Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/KR05/000668

International filing date: 09 March 2005 (09.03.2005)

Document type: Certified copy of priority document

Document details: Country/Office: KR
Number: 10-2004-0019066
Filing date: 16 March 2004 (16.03.2004)

Date of receipt at the International Bureau: 30 June 2005 (30.06.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

대한민국 특허청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office

출 원 번 호 Application Number	:	특허출원 2004년 제 0019066 호 10-2004-0019066
출 원 일 자 Date of Application	:	2004년 03월 16일 MAR 16, 2004
출 원 인 Applicant(s)	:	양세양 YANG, Sei Yang

2005 년 06 월 09 일

특 허 청

COMMISSIONER

【서지사항】

【서류명】 특허출원서

【권리구분】 특허

【수신처】 특허청장

【참조번호】 0001

【제출일자】 2004.03.16

【발명의 국문명칭】 다양한 검증 플랫폼들의 통합 사용을 지원하는 검증 장치 및 이를 이용한 검증 방법

【발명의 영문명칭】 Verification Apparatus Supporting the Use of Unifying Different Verification Platforms, and the Verification Method Using the Same

【출원인】

【성명】 양세양

【출원인코드】 4-1998-037998-4

【발명자】

【성명】 양세양

【출원인코드】 4-1998-037998-4

【우선권 주장】

【출원국명】 KR

【출원종류】 특허

【출원번호】 10-2001-0057742

【출원일자】 2001.09.14

【증명서류】 미첨부

【취지】 특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 출원인 양세양 (인)

【수수료】

【기본출원료】	20 면	39,000 원
【가산출원료】	67 면	227,800 원
【우선권주장료】	0 건	0 원
【심사청구료】	0 항	0 원
【합계】	266,800 원	
【감면사유】	개인(70%감면)	
【감면후 수수료】	80,040 원	
【첨부서류】	1.요약서 · 명세서(도면)_2통	

【요약서】

【요약】

본 발명은 설계된 매우 복잡한 디지털 시스템의 설계 검증을 위한 검증 장치와 이를 이용한 효과적인 검증 방법에 관한 것이다.

본 발명에서는 임의의 컴퓨터에서 수행되어지는 본 발명의 검증 소프트웨어로 하여금 설계 코드에 부가적인 코드를 부가하여 다양한 검증 플랫폼들을 함께 사용하여 효과적인 검증을 신속하게 수행하는 것을 가능하게 한다. 1 이상의 시뮬레이터, 1 이상의 정식검증 툴, 1 이상의 시뮬레이션 가속기, 1 이상의 하드웨어 에뮬레이터, 1 이상의 프로토타이핑시스템 등과 같은 여러 가지 다른 종류의 검증 플랫폼들을 사용하여 설계 검증을 수행함에 있어서 각기 상이한 성격들을 갖는 여러 가지의 검증 플랫폼들을 통합된 방식으로 사용되어지게 함으로서 각 검증 플랫폼들의 이용률을 극대화함과 동시에 전체의 검증 속도를 증가시키고 신속한 디버깅을 가능하게 한다.

【대표도】

도 1

【명세서】

【발명의 명칭】

다양한 검증 플랫폼들의 통합 사용을 지원하는 검증 장치 및 이를 이용한 검증 방법{Verification Apparatus Supporting the Use of Unifying Different Verification Platforms, and the Verification Method Using the Same}

【도면의 간단한 설명】

<1> 도1 (a)는 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면.

<2> 도1 (b)에서부터 도 1(s)까지는 본 발명에 관한 설계 검증 장치의 또 다른 일례들을 개략적으로 도시한 도면.

<3> 도2 는, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 1 이상의 시뮬레이터들로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면.

<4> 도3 은, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 1 이상의 모델검사기 내지는 특성검사기들로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면.

<5> 도4 는, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드

웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 시뮬레이터와 모델검사기 내지는 특성검사기를 같이 사용하여 병렬적으로 수행하는 과정을 개략적으로 도시한 도면.

<6> 도5 는, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 하나의 시뮬레이터로 순서적으로 수행하는 과정을 개략적으로 도시한 도면.

<7> 도6 은, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 하나의 모델검사기 내지는 특성검사기로 순서적으로 수행하는 과정을 개략적으로 도시한 도면.

<8> 도7 은, 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 FPGA 보드로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면.

<9> 도8 은 본 발명에서의 검증 장치를 이용하여 설계에 존재하는 1 이상의 설계 오류를 발견하여 수정하는 과정을 개략적으로 도시한 순서도 도면.

<10> 도9 는 본 발명에서의 검증 장치를 이용하여 설계 검증을 수행하는 과정을 개략적으로 도시한 순서도 도면.

<11> <도면의 주요부분에 대한 부호의 설명>

<12> 28 : 프로토타이핑시스템의 시스템소프트웨어 컴퍼넌트

<13> 29 : 시뮬레이션가속기의 시스템소프트웨어 컴퍼넌트

<14> 30 : 하드웨어에뮬레이터의 시스템소프트웨어 컴퍼넌트

<15> 31 : 검증 소프트웨어에서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 추가 코드나 추가 회로를 자동적으로 부가하고 검증 준비를 진행하는 소프트웨어모듈

<16> 32 : 모델검사기 또는 특성 검사기 33 : 시뮬레이터

<17> 34 : 검증 소프트웨어에서 검증 실행 도중에 1 이상의 컴퓨터간에 파일이나 데이터 전송을 가능하게 하고, 1차 검증 실행을 진행하고 1차 이후의 검증 실행을 위한 준비를 진행하고 1차 이후의 검증 실행을 진행시키는 소프트웨어모듈

<18> 35 : 컴퓨터

<19> 36 : 하드웨어에뮬레이터 플랫폼의 하드웨어 컴퍼넌트

<20> 37 : 시뮬레이션가속기 플랫폼의 하드웨어 컴퍼넌트

<21> 38 : 프로토타이핑시스템 플랫폼의 하드웨어 컴퍼넌트

<22> 39 : 하드웨어기반의 검증 플랫폼의 시스템소프트웨어 컴퍼넌트

<23> 40 : 프로토타이핑시스템의 일종인 임의의 FPGA 보드

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<24> 본 발명은 설계된 수백만 게이트급 이상의 디지탈 시스템의 설계를 검증하는 기술에 관한 것으로, 설계된 수백만 게이트급 이상의 디지탈 시스템을 시뮬레이션, 정식검증, 시뮬레이션 가속, 하드웨어 에뮬레이션, 프로토타이핑들을 통하여 검증하고자 하는 경우에 검증의 성능와 효율성을 증가시키는 검증 장치 및 이를 이용한 검증 방법에 관한 것이다.

<25> 최근에 집적회로의 설계 및 반도체 공정기술이 급격하게 발달함에 따라 디지탈 회로 설계의 규모가 최소 수백만 게이트급에서 수천만 게이트급까지 커짐은 물론 그 구성이 극히 복잡해지고 있는 추세이고, 이와 같은 추세는 계속적으로 확대되고 있는 추세로 가까운 미래에 일억 게이트급 이상의 설계도 예상되고 있다. 그러나, 시장에서의 경쟁은 더욱 더 치열해지므로 빠른 시간 내에 우수한 제품을 개발하여야만 함으로 빠른 시간 내에 자동화된 방법으로 설계된 회로를 효율적으로 설계 검증하기 위한 효과적인 방법의 필요성이 더욱 커지고 있다.

<26> 지금까지는 설계된 디지탈 회로를 설계 검증하기 위하여서 하드웨어 기술언어(Hardware Description Language, 앞으로 이를 HDL로 약칭함)를 이용한 설계 초기에는 소프트웨어적 접근법인 HDL 시뮬레이터들(예로 Verilog 시뮬레이터, VHDL 시뮬레이터, 또는 SystemC 시뮬레이터 등)이 주로 사용되어지고 있다. 이와 같은 시뮬레이터는 설계 검증 대상회로를 소프트웨어적으로 모델링한 순차적인 인스트럭션 시퀀스로 구성된 소프트웨어 코드를 컴퓨터 상에서 순차적으로 수행하여야 함으로 상기 수백만 게이트급 이상의 설계에 대해서는 시뮬레이션 성능의 저하가 설계

대상의 크기에 비례하여 발생하는 것이 문제가 되고 있다. 일 예로 시스템온칩(System On a Chip, 앞으로는 이를 SOC로 약 칭함) 설계에서와 같이 1000만 게이트급 이상의 설계를 HDL 시뮬레이터로 시뮬레이션하는 경우에 현존하는 제일 빠른 프로세서를 장착한 컴퓨터에서 해당 HDL 시뮬레이터로 설계를 시뮬레이션하는 경우에 시뮬레이션 속도는 레지스터전송수준(Register Transfer Level, 앞으로 이를 RTL로 약칭함)으로 하는 경우에 10 cycles/sec를 넘기기 어려우며, 게이트 수준에서 시뮬레이션을 진행하면 1-2 cycles/sec를 넘기기가 어려운 것이 매우 일반적이다. 그러나 해당 설계를 검증하고자 필요한 시뮬레이션 사이클은 최소 수백만 사이클에서부터 최대 수십억 사이클이 필요함으로 전체 시뮬레이션 시간은 상상을 초월하게 된다. 뿐만 아니라, 최근에는 이와 같은 복잡한 디지털 시스템 설계를 시뮬레이션을 이용하여 검증하는 경우에 모든 설계 오류들을 발견하여 제거할 수 있는 테스트벤치를 작성하는 것에 대한 어려움이 증대하고 있다.

<27> 이와 같은 장시간의 검증 시간의 단축과 테스트벤치 작성의 어려움을 해소시키기 위하여 현재 사용되는 기법들은 다음과 같은 것들이 있는데, 첫째는 하드웨어 기반의 검증 시스템(예로 시뮬레이션가속기, 하드웨어 에뮬레이터, 프로토타이핑시스템 등)을 사용하거나, 둘째는 특성검사(property checking) 혹은 모델검사(model checking) 내지는 등가검사(equivalence checking)과 같은 정식검증(formal verification)을 사용하는 것이다. 그러나 하드웨어 기반의 검증 시스템을 사용하는 것은 설계 초기에는 적용이 불가능하고, 합성이나 컴파일 과정이 HDL 시뮬레이티를 사용하는 것보다 오래 걸리며, 사용하기가 HDL 시뮬레이터에 비하여 매우 어

렵고, 시스템의 구입 비용과 유지/보수 비용이 매우 클 뿐만 아니라, 무엇보다도 설계자나 검증엔지니어들이 HDL 시뮬레이터에 대한 선호도가 이들 하드웨어 기반의 검증 시스템에 비하여 매우 높고, HDL 시뮬레이터로는 아무 문제 없이 수행이 되는 설계 코드들이 하드웨어 기반의 검증 시스템에서는 수행되지 않는 경우가 많아서 이들 하드웨어 기반의 검증 시스템들은 제한적인 상황과 제한적인 사용자들에서만 사용되고 있다. 뿐만 아니라, 이들 하드웨어기반의 검증시스템들은 시뮬레이터와 같은 수준의 제어도(controllability)와 관측도(observability)를 제공하지 못하고 설계검증대상(DUV: Design Under Verification, 이 후로는 DUV로 약칭함)에 존재하는 모든 신호선들을 탐침하게 되면 수행속도가 200% 이상 늘어나는 문제점도 가지고 있다. 또한 정식검증 툴을 이용하는 것은 등가검사이외에 특성검사나 모델검사를 적용하기 위해서는 이를 위한 추가적인 작업을 요구하게 됨으로 설계자들에게 새로운 부담을 주게 될 뿐만 아니라, 이와 같은 정식검증 기법은 검증 대상이 되는 설계의 크기가 커지게 되면 해당 기법이 상태 폭발(state explosion)의 문제로 인하여 컴퓨터 상에서 수행이 불가능해진다는 치명적인 약점이 있다.

<28> 또 다른 문제점으로는 검증을 하기 위하여 설계 코드를 이용한 검증 실행이 일정 검증 사이클 동안(예로 시뮬레이션으로 검증을 수행하는 경우에 0 나노초 시뮬레이션 사이클에서부터 10,000,000 나노초 시뮬레이션 사이클까지) 이루어져야 하는데 이 과정에서 설계 코드의 수행이 설계자가 의도하는 대로 진행되는 가를 확인하는 과정이 반드시 필요하다. 이와 같은 확인 과정에서 필요한 것이 설계 코드에서 존재하는 시그널들이나 변수들에 대한 탐침(probing)인데, 이와 같은 탐침을

통하여 설계에 대한 가시도(visibility)가 확보되는 것이 설계 코드에 존재하는 오류들의 위치를 확인하고 제거하는 것에 반드시 필요하게 된다. 그러나 이와 같은 탐침에서의 문제점으로는 설계 코드에는 매우 많은 수의 시그널들이나 변수들이 존재하게 되는데, 이들 중에서 상기의 일정 검증 사이클 동안에서 설계 코드를 이용한 검증실행이 실제 수행되기 전에 설계 오류의 위치를 알아내고 이를 제거하기 위해서 탐침이 필요한 시그널들이나 변수들을 선정하는 것이 매우 어렵다는 것이다. 현재 사용되는 방법은 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑 실행 전에 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침 대상으로 선정한 후에 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것이 한 방법이다. 그러나, 이와 같이 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침대상으로 하면, 탐침대상을 정하지 않고 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것에 비하여 실행 속도가 최소 2배에서 최대 5배 이상 증가하게 되는 문제점이 있다.

<29> 또 다른 문제점으로는 수백만 게이트급 이상의 디지털 시스템을 효율적으로 검증하기 위해서는 시뮬레이션, 시뮬레이션가속, 에뮬레이션, 프로토타이핑, 정식검증들 중에서 두 개 이상의 검증기술을 같이 사용한 검증 수행 과정이 요구되는데, 실제 이들 기술들을 같이 사용하는 경우에도 단지 하나의 기술(예로 시뮬레이션)을 사용하여 최대한 설계 오류들을 찾아서 수정한 후에, 또 다른 기술(예로 정식검증)을 사용하여 남아있는 설계 오류들을 찾아서 수정하는 단순한 수준

에서 두 개 이상의 검증기술을 같이 사용하는 수준에 머물고 있다.

【발명이 이루고자 하는 기술적 과제】

<30> 따라서, 본 발명의 목적은 초대규모급 디지털시스템 설계에 대한 검증을 위한 검증의 성능 향상과 효율성을 증대시키는 검증 장치 및 이를 이용한 검증 방법을 제공함에 있다. 구체적으로는 시뮬레이션, 정식검증, 시뮬레이션가속, 하드웨어에뮬레이션, 프로토타이핑 등을 강하게 통합된 환경에서 혼합적으로 사용함으로서 검증의 속도 향상과 더불어 효율성을 증대시키고, 탐침을 위한 가시도를 제공함에 있어서 검증의 성능 저하를 유발시키지 않고, 설계 오류들에 대한 신속한 발견과 디버깅을 가능하게 한다.

【발명의 구성】

<31> 상기 목적들을 달성하기 위하여, 본 발명에 따른 검증 장치는 검증 소프트웨어와, 1 이상의 HDL 시뮬레이터 내지는 1 이상의 정식검증툴이 인스톨된 1이상의 컴퓨터와, 1 이상의 하드웨어에뮬레이터 내지는 1 이상의 시뮬레이션가속기 내지는 1 이상의 프로토타이핑시스템으로 구성된다. 또는 본 발명에 따른 검증 장치는 검증 소프트웨어와, 1 이상의 HDL 시뮬레이터가 인스톨된 1 이상의 컴퓨터와 1 이상의 정식검증툴이 인스톨된 1이상의 컴퓨터와, 1 이상의 하드웨어에뮬레이터 내지는 1이상의 시뮬레이션가속기 내지는 1 이상의 프로토타이핑시스템으로 구성된다. 본 발명의 검증 소프트웨어는 컴퓨터에서 실행되며, 만일 상기 설계 검증 장치에 2이상의 컴퓨터들이 있는 경우에는 이들 2이상의 컴퓨터는 네트워크로 연결되어져서

컴퓨터들 간에 파일들의 이동도 가능하게 한다.

<32> 본 발명에서 제안되는 검증 장치와 검증 방법은 설계 코드 자체를 검증하는 함수적 검증(functional verification)에 사용될 수 있을 뿐만 아니라, 설계 코드를 합성한 게이트수준의 네트리스트를 이용한 게이트수준의 검증에서도 사용될 수 있고, 또는 배치(placement) 및 배선(routing)이 되고 추출된 타이밍정보를 게이트수준의 네트리스트에 첨부시켜(back-annotated) 수행하는 타이밍 검증에서도 사용될 수 있다. 그러나 앞으로의 설명은 설계 코드 자체를 검증하는 함수적 검증에 대하여 하기로 하며, 게이트수준 검증이나 타이밍 검증에 대해서도 같은 방법을 적용할 수 있음으로 구체적 설명은 생략하기로 한다.

<33> 검증 소프트웨어는 HDL(예로 Verilog, VHDL, System Verilog, System C등)을 사용하여 작성된 설계 코드를 읽은 후에 여기에다 추가적인 코드를 부가한다. 부가된 코드는 상기 설계 코드를 검증하기 위한 검증 실행 과정을 시뮬레이션가속 내지는 하드웨어에뮬레이션 내지는 프로토타이핑 내지는 시뮬레이션 내지는 정식검증의 다른 검증 기법들을 1회 이상 혼합하여 수행되는 과정을 자동화된 방식으로 일어날 수 있도록 하는데, 검증 소프트웨어는 이외의 검증 실행을 위한 준비(예로 컴파일, 로딩 등)를 진행하고, 검증 실행도 전체적으로 제어하는 역할을 수행한다. 상기 다른 검증 기법들을 1회 이상 혼합하여 수행되는 과정이란, 다른 검증 기법들을 독립적으로 사용하는 기존의 검증 방법들과는 달리 하나의 검증 기법(예로 시뮬레이션)을 수행하여 얻어진 검증 결과를 다른 검증 기법(예로 하드웨어에뮬레이션)에서 사용하여 검증을 효과적으로 수행되게 하는 것을 말한다. 구체적으로는 검증을 하

기 위하여 특정 검증 플래폼(예로 HDL 시뮬레이터) 상에서 설계 코드를 이용한 검증 실행이 일정 검증 사이클 동안(예로 시뮬레이션으로 검증을 수행하는 경우에 0 나노초 시뮬레이션 사이클에서부터 10,000,000 나노초 시뮬레이션 사이클까지) 이루어져야 하는데 이 과정에서 검증 플랫폼(시뮬레이터 내지는 시뮬레이션가속기 내지는 하드웨어에뮬레이터 내지는 프로토타이핑시스템 내지는 정식검증툴와 같이 설계검증을 수행하기 위하여 설계 코드나 해당 RTL/gate-level 네트리스트/함수적등가의 다른 코드를 읽어들여 컴파일하여 실행하는 소프트웨어나 하드웨어 시스템) 상에서 설계 코드의 수행이 설계자가 의도하는 대로 진행되는 가를 확인하는 과정이 반드시 필요하다. 검증 플랫폼이 시뮬레이션가속기 내지는 하드웨어에뮬레이터 내지는 프로토타이핑시스템과 같은 하드웨어기반의 검증 시스템(hardware-assisted verification system)의 경우에는 해당 검증 플랫폼은 하드웨어 컴퍼넌트와 시스템 소프트웨어 컴퍼넌트로 구성되어 있다. 이와 같은 확인 과정에서 필요한 것이 설계 코드에서 존재하는 시그널들이나 변수들에 대한 탐침(probing)이다. 이와 같은 탐침을 통하여 설계에 대한 가시도(visibility)가 확보되는 것이 설계 코드에 존재하는 오류들의 위치를 확인하고 제거하는 것에 반드시 필요하게 된다. 그러나 이와 같은 탐침에서의 문제점으로는 설계 코드에는 매우 많은 수의 시그널들이나 변수들이 존재하게 되는데 이들 중에서 상기의 일정 검증 사이클 동안에서 설계 코드를 이용한 검증 실행이 실제 수행되기 전에 설계 오류의 위치를 알아내고 이를 제거하기 위해서 탐침이 필요한 시그널들이나 변수들을 선정하는 것이 매우 어렵다는 것이다. 현재 사용되는 방법은 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션

내지는 프로토타이핑 실행 전에 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정한 후에 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것이 한 방법이다. 그러나, 이와 같이 검증 대상에 존재하는 모든 시그널들과 변수들을 탐침대상으로 하면, 탐침대상을 정하지 않고 상기 시뮬레이션 내지는 시뮬레이션가속 내지는 에뮬레이션 내지는 프로토타이핑을 실행하는 것에 비하여 실행 속도가 최소 2배에서 최대 5배 이상 증가하게 되는 문제점이 있다.

<34> 기존의 설계 검증을 위한 검증 실행이 하나의 검증 플랫폼만을 반복적으로 이용하여 이루어지는 것과는 달리 본 발명에서의 검증 실행은 2 이상의 상이한 검증 플랫폼을 번갈아 이용하여 어루어진다. 이를 위하여 본 발명에서 제안하는 검증 장치는 검증 소프트웨어로써 설계 코드에 자동화된 방식을 통하여 부가 코드를 부가하여 특정 검증 플랫폼 상에서 1차 검증 실행을 수행하면서 1차 시뮬레이션 이후에 다른 검증 플랫폼 상에서 수행되는 검증 실행을 특정 검증사이클 구간들이나 설계 코드에 존재하는 특정 블록들에 한정하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할 수 있도록 하고, 상기 1차 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 상기 최소한의 정보를 수집하고, 이 수집된 정보를 이용하여 다른 검증 플랫폼 상에서 1차 이후의 검증 실행을 신속하게 수행하는 것을 가능하게 한다. 1차 이후의 검증 실행을 신속하게 수행하기 위해서, 필요 시에 상기 1차 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 수집된 정보를 이용하여서 1차 이후의 검증 실행을 1 이상의 검증

플랫폼을 이용하여 병렬적으로 수행하거나 혹은 하드웨어기반의 검증 플랫폼(예로 FPGA 보드)을 이용하여 수행하거나 혹은 검증 코드에 존재하는 특정 블록들만을 한정하여 수행함으로서 신속한 검증 실행을 가능하게 한다.

<35> 다음에는 본 발명에서의 검증 장치와 검증 방법들에 대한 구체적인 사례들을 예시하면서 설명하기로 한다.

<36> 첫 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 시뮬레이터로 구성되고, 상기 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 하드웨어에뮬레이터로 하고 하드웨어에뮬레이터를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 시뮬레이터를 사용하여 진행한다. 이와 같은 방법의 장점은 수십억원 이상으로 매우 고가인 하드웨어에뮬레이터의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 하드웨어에뮬레이터의 이용률을 높일 수 있는 것은, 하드웨어에뮬레이터만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 하드웨어에뮬레이션을 진행하게 되면 에뮬레이션의 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 하드웨어에뮬레이터를 사용하여 진

행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 에뮬레이션의 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 하드웨어에뮬레이터를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 하드웨어에뮬레이터의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 시뮬레이터를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 시뮬레이터로 사용하여 시뮬레이션을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 시뮬레이터를 이용하여 2 이상의 시뮬레이션을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 시뮬레이션의 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 하드웨어에뮬레이터를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(에뮬레이션 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 정보이다. 여기에서 DUV의 상태정보란 임의의 검증실행 시점에서의 DUV 내에 존재하는 모든 메모리소자(플립플롭이나 래치나 RAM 이나 ROM 그리고 조합궤환루프(combinational feedback)가 있는 논

리게이트 등의 기억소자)의 값을 가르킨다. 그리고 SOC와 같은 매우 복잡한 시스템에 대용량의 메모리(예로 SDRAM이나 DDR-RAM)가 존재하는 경우에는 이와 같은 메모리들을 제외한 나머지 설계 대상을 DUV로 선정하는 것도 가능하다. 이와 같은 경우에는 상기 검증 대상에서 제외된 메모리에서 나머지 설계대상으로 인가되는 신호선들도 DUV의 입력이나 입출력이 될 것임으로 이와 같은 신호선들로 상기 검증 실행 전체구간에 대하여 탐침을 수행하는 대상으로 선정되어야 한다(이 정보를 시간병렬가능정보라 칭하고, 이를 앞으로 사용함). DUV의 상태정보를 검증 실행 도중의 임의의 시점에서 저장하는 것은 현재 사용 중인 하드웨어에뮬레이터들(예로 Cadence의 Palladium이나 Cobalt, Mentor의 Celaro나 V-Station, Axis의 Extreme-II 등) 대부분이 DUV 내부에 존재하는 어떠한 신호선들도 탐침할 수 있음으로 문제없이 가능하다. 또한 DUV의 모든 입력과 입출력들을 검증 실행 전체 구간에 걸쳐서 탐침하여 저장시키는 것은 하드웨어에뮬레이터들에 존재하는 대용량의 메모리를 이용하거나 또는 하드웨어에뮬레이터 플랫폼과 연결되어 같이 사용되는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다. 상태정보를 이용하면 1차 이후의 검증 실행을 반드시 1차 검증 실행의 맨 시작 시점(앞으로는 이를 "0 검증사이클 시간"이라 약하여 칭함)에서부터 시작하지 않아도 되는 매우 중요한 장점이 있다. 또 다른 하나는 1차 검증 실행(에뮬레이션 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 정보이다(이 정보를 공간병렬가능정보라 칭하고, 이를 앞으로 사용함). 또한 DUV에 존재하는 2 이

상들의 블록들 각각에 존재하는 모든 입력과 입출력들을 검증 실행 전 구간에 걸쳐서 탐침하여 저장시키는 것은 하드웨어에뮬레이터들에 존재하는 대용량의 메모리를 이용하거나 또는 하드웨어에뮬레이터 플랫폼과 연결되어 같이 사용되는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다.

<37> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들(0, T_1), (T_1, T_2), ...(T_{n-1}, T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터를 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 에뮬레이션을 통한 검증 실행에서 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 시뮬레이션에서 2 이상의 시뮬레이터를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1}, T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<38> 두 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 시뮬레이터로 구성되고, 상기 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 시뮬레이션가속기로 하여 이를 사용하여 1차 검증 실행을 수행

하고, 1차 이후의 검증 실행을 1 이상의 시뮬레이터를 사용하여 진행한다. 이와 같은 방법의 장점은 수억원 이상으로 매우 고가인 시뮬레이션가속기의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 시뮬레이션가속기의 이용률을 높일 수 있는 것은, 시뮬레이션가속기만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션가속을 진행하게 되면 시뮬레이션가속의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증실행을 시뮬레이션가속기를 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 시뮬레이션가속의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 시뮬레이션가속기를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 시뮬레이션가속기의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 시뮬레이터를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 시뮬레이터로 사용하여 시뮬레이션을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다.

<39> 한가지 사용할 수 있는 방법은 2 이상의 시뮬레이터를 이용하여 2 이상의 시뮬레이션을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 시뮬레이션의 결과를

통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 시뮬레이션가속기를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수행하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(시뮬레이션가속 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. SOC과 같은 매우 복잡한 시스템에 대용량의 메모리(예로 SDRAM이나 DDR-RAM)가 존재하는 경우에는 이와 같은 메모리들을 제외한 나머지 설계 대상을 DUV로 선정하는 것도 가능하다. 이와 같은 경우에는 상기 검증 대상에서 제외된 메모리에서 나머지 설계대상으로 인가되는 신호선들도 DUV의 입력이나 입출력이 될 것임으로 이와 같은 신호선들로 상기 검증 실행 전체 구간에 대하여 탐침을 수행하는 대상으로 선정되어야 한다. DUV의 상태정보를 검증 실행도중의 임의의 시점에서 저장하는 것은 현재 사용 중인 시뮬레이션가속기들(예로 Mentor의 NSIM 또는 ARES, Axis의 Excite-Ⅱ, Tharas의 Hammer, Pittsburgh Simulation의 V400 등) 대부분이 DUV 내부에 존재하는 어떠한 신호선들도 탐침할 수 있음으로 문제없이 가능하다. 또한 DUV의 모든 입력과 입출력들을 검증 실행 전체 구간에 걸쳐서 탐침하여 저장시키는 것은 시뮬레이션가속기들에 존재하는 대용량의 메모리를 이용하거나 또는 시뮬레이션가속기 플랫폼과 연결되어 같이 사용되

는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다. 또 다른 하나는 1차 검증 실행(시뮬레이션가속 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다. DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들을 검증 실행 전 구간에 걸쳐서 탐침하여 저장시키는 것은 시뮬레이션가속기들에 존재하는 대용량의 메모리를 이용하거나 또는 시뮬레이션가속기 플랫폼과 연결되어 같이 사용되는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다.

<40> 시간병렬가능정보는 1차 검증 실행 구간(0, T)를 2 이상의 구간들(0, T_1), (T_1, T_2), ... (T_{n-1}, T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터를 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 시뮬레이션가속을 통한 검증 실행에서 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 시뮬레이션에서 2 이상의 시뮬레이터를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1}, T)와 B_1, B_2,

... B_n 모두에서 병렬적인 수행도 가능하다.

<41> 세 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 프로토타이핑시스템과 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 시뮬레이터로 구성되고, 상기 1 이상의 프로토타이핑시스템과 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 프로토타이핑시스템으로 하여 이를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 시뮬레이터를 사용하여 진행한다. 이와 같은 방법의 장점은 프로토타이핑시스템의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 프로토타이핑시스템의 이용률을 높일 수 있는 것은, 프로토타이핑시스템만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션가속을 진행하게 되면 프로토타이핑시스템의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다(예로 FPGA를 채용한 프로토타이핑시스템의 경우에 Xilinx FPGA의 read-back 기능을 이용한 탐침으로 가시도를 얻고자 하는 경우에는 실행 속도가 100,000배 이상 저하됨). 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 프로토타이핑시스템을 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침

에 의한 프로토타이핑 시스템의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 프로토타이핑시스템을 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 프로토타이핑시스템의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 시뮬레이터를 사용하여 진행된다. 1차 이후의 검증실행을 1 이상의 시뮬레이터로 사용하여 시뮬레이션을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 시뮬레이터를 이용하여 2 이상의 시뮬레이션을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 시뮬레이션의 결과를 통합하여 전체검증 실행의 결과로 이용할 수 있다. 이를 위해서 프로토타이핑시스템을 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(프로토타이핑시스템 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. SOC과 같은 매우 복잡한 시스템에 대용량의 메모리(예로 SDRAM이나 DDR-RAM)가 존재하는 경우에는 이와 같은 메모리들을 제외한 나머지 설계 대상을 DUV로 선정하는 것도 가능하다. 이와 같은 경우에는 상기 검증대상에서 제외된 메모리에서 나머지 설계대상으로 인가되

는 신호선들도 DUV의 입력이나 입출력이 될 것임으로 이와 같은 신호선들로 상기 검증 실행 전체 구간에 대하여 탐침을 수행하는 대상으로 선정되어야 한다. DUV의 상태정보를 검증 실행도중의 임의의 시점에서 저장하는 것은 현재 사용 중인 프로토타이핑시스템들(예로 Aptix의 System Explorer 또는 Pathfinder IP Validation Station와 같은 상요 프로토타이핑시스템 뿐만 아니라, 설계자나 검증 엔지니어가 직접 제작하는 임의의 FPGA 기반의 프로토타이핑보드 등 모두를 프로토타이핑시스템이라 할 수 있다) 대부분이 DUV 내부에 존재하는 어떠한 신호선들도 탐침할 수 있음으로 문제없이 가능하다(예로 FPGA의 read-back 기술을 이용하거나 미국특허 US6,701,491에서 제안된 기술을 사용하면 됨). DUV의 모든 입력과 입출력들을 검증 실행 전체 구간에 걸쳐서 탐침하여 저장시키는 것은 프로토타이핑시스템들에 존재하는 대용량의 메모리를 이용하거나 또는 프로토타이핑시스템의 플랫폼과 연결되어 같이 사용되는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다. 또다른 하나는 1차 검증 실행(프로토타이핑시스템 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다. DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들을 검증 실행 전 구간에 걸쳐서 탐침하여 저장시키는 것은 프로토타이핑시스템들에 존재하는 대용량의 메모리를 이용하거나 또는 프로토타이핑시스템의 플랫폼과 연결되어 같이 사용되는 컴퓨터에 존재하는 주메모리를 이용하거나 주메모리와 하드디스크를 이용할 수 있다.

<42> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1}, T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 시뮬레이션 실행에서 2 이상의 시뮬레이터를 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 프로토타이핑시스템 수행을 통한 검증 실행에서 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 시뮬레이션에서 2 이상의 시뮬레이터를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1}, T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<43> 이와 같은 3가지 사례들에서 1차 이후의 시뮬레이션을 2 이상의 시뮬레이터로 병렬적으로 수행하는 것이 유리하지만, 만일 사용가능한 시뮬레이터가 하나인 경우에는 2 이상의 시뮬레이션들을 병렬적으로 수행하는 대신에 순차적으로 수행하는 것도 가능하다. 이와 같은 순차적인 수행 과정에서는 (0, T_1), (T_1, T_2), ... (T_{n-1}, T)으로 시간병렬가능정보가 구성된 경우에는 검증시간에서 뒤에 있는 (T_{n-1}, T)부터 (T_{n-1}, T_{n-2})... (T_1, T_2), 그리고 제일 앞에 있는 (0, T_1) 등의 순서를 정하여 진행하면서 진행 도중에 설계 오류의 위치와 원인이 밝혀지게 되면 이 이후의 시뮬레이션을 생략하는 것이 가능하다.

<44> 네 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 정식검증 툴(모델검사기 내지는 특성검사기)로 구성되고, 상기 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 하드웨어에뮬레이터로 하여 하드웨어에뮬레이터를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 정식검증 툴을 사용하여 진행한다. 이와 같은 방법의 장점은 수십억원 이상으로 매우 고가인 하드웨어에뮬레이터의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 하드웨어에뮬레이터의 이용률을 높일 수 있는 것은, 하드웨어에뮬레이터만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 하드웨어에뮬레이션을 진행하게 되면 에뮬레이션의 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 하드웨어에뮬레이터를 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 에뮬레이션의 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 하드웨어에뮬레이터를 동일설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 하드웨어에뮬레이터의 이용률을 높일 수 있게 한다. 1차 이후의

검증 실행은 1이상의 정식검증 툴을 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 정식검증 툴로 사용하여 모델 검사나 특성 검사 과정을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 정식검증 툴을 이용하여 2 이상의 모델 검사나 특성 검사를 병렬적으로 진행하고, 필요시에는 이들 2 이상의 모델 검사나 특성 검사의 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 하드웨어에뮬레이터를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 정식검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보는 DUV에 존재하는 1 이상의 상태정보이다. 이는 1차 검증 실행(에뮬레이션 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한 번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하여 얻어지는 시간병렬가능정보이다.

<45> 시간병렬가능정보는 1차 검증 실행 구간 (0, T_n)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1}, T_n)로 나누어서 0, T_1, T_2, ... T_{n-1} , T_n 시점들 각각에서의 상태들 S_0, S_1, S_2, ... S_{n-1}, S_n 들을 모두 이용하거나 또는 이들 중에서 특정 상태들을 선택적으로 이용하여 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩으로 초기상태로 설정하여 병렬적으로 수행되어지는 것이 가능하다. 뿐만 아니라 DUV가 B_1, B_2, ... B_n과 같이 1이상의 블록들로 나누어져서

구성되어 있는 경우에 상기 특정 검증시점에서의 DUV의 특정 상태 S_i를 이들 블록별로 나누어서 S_{i1}, S_{i2}, ... S_{in}의 1 이상의 블록상태로 재구성하여 이들 각각의 블록상태들로 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩 초기상태로 설정하여 블록별로 병렬적으로 수행되어지는 것도 가능하다.

<46> 다섯 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1이상의 정식검증 툴(모델검사기 내지는 특성검사기)로 구성되고, 상기 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 시뮬레이션가속기로 하여 시뮬레이션가속기를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1이상의 정식검증 툴을 사용하여 진행한다. 이와 같은 방법의 장점은 수억원 이상으로 매우 고가인 시뮬레이션가속기의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 시뮬레이션가속기의 이용률을 높일 수 있는 것은, 시뮬레이션가속기만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션가속을 진행하게 되면 시뮬레이션가속의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 시뮬레이션가속기를 사

용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 시뮬레이션가속의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 시뮬레이션가속기를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 시뮬레이션가속기의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 정식검증 툴을 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 정식검증 툴로 사용하여 모델 검사나 특성 검사 과정을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 정식검증 툴을 이용하여 2 이상의 모델 검사나 특성 검사를 병렬적으로 진행하고, 필요시에는 이들 2 이상의 모델 검사나 특성 검사의 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 시뮬레이션가속기를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 정식검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보는 DUV에 존재하는 1 이상의 상태정보이다. 이는 1차 검증 실행(시뮬레이션가속 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하여 얻어지는 시간병렬가능정보이다.

<47> 시간병렬가능정보는 1차 검증 실행 구간 (0, T_n)를 2 이상의 구간들(0, T

$_1$), (T_1, T_2), ... (T_{n-1}, T_n)로 나누어서 0, T_1, T_2, ... T_{n-1} , T_n 시점들 각각에서의 상태들 S_0, S_1, S_2, ... S_{n-1}, S_n 들을 모두 이용하거나 또는 이들 중에서 특정 상태들을 선택적으로 이용하여 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩으로 초기상태로 설정하여 병렬적으로 수행되어지는 것이 가능하다. 뿐만 아니라 DUV가 B_1, B_2, ... B_n과 같이 1이상의 블록들로 나누어져서 구성되어 있는 경우에 상기 특정 검증시점에서의 DUV의 특정 상태 S_i를 이들 블록별로 나누어서 S_{i1}, S_{i2}, ... S_{in}의 1 이상의 블록상태로 재구성하여 이들 각각의 블록상태들로 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩 초기상태로 설정하여 블록별로 병렬적으로 수행되어지는 것도 가능하다.

<48> 여섯 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 프로토타이핑시스템과 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 정식검증 툴(모델검사기 내지는 특성검사기)로 구성되고, 상기 1 이상의 프로토타이핑시스템과 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 프로토타이핑시스템으로 하여 프로토타이핑시스템을 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 정식검증 툴을 사용하여 진행한다. 이와 같은 방법의 장점은 프로토타이핑시스템의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 프로토타이핑시스템의 이용률을 높일 수 있는 것은, 프로토타이핑시

스템만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 프로토타이핑시스템의 수행을 진행하게 되면 프로토타이핑의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 프로토타이핑시스템을 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 프로토타이핑시스템의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 프로토타이핑시스템을 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 프로토타이핑시스템의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1이상의 정식검증 툴을 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 정식검증 툴로 사용하여 모델 검사나 특성 검사 과정을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 정식검증 툴을 이용하여 2 이상의 모델 검사나 특성검사를 병렬적으로 진행하고, 필요시에는 이들 2 이상의 모델 검사나 특성 검사의 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 프로토타이핑시스템을 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 정식검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의

정보는 DUV에 존재하는 1 이상의 상태정보이다. 이는 1차 검증 실행(프로토타이핑 시스템실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하여 얻어지는 시간병렬가능정보이다.

<49> 시간병렬가능정보는 1차 검증 실행 구간 (0, T_n)를 2 이상의 구간들(0, T_1), (T_1, T_2), ... (T_{n-1}, T_n)로 나누어서 0, T_1, T_2, ... T_{n-1} , T_n 시점들 각각에서의 상태들 S_0, S_1, S_2, ... S_{n-1}, S_n 들을 모두 이용하거나 또는 이들 중에서 특정 상태들을 선택적으로 이용하여 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩으로 초기상태로 설정하여 병렬적으로 수행되어지는 것이 가능하다. 뿐만 아니라 DUV가 B_1, B_2, ... B_n과 같이 1이상의 블록들로 나누어져서 구성되어 있는 경우에 상기 특정 검증시점에서의 DUV의 특정 상태 S_i를 이들 블록별로 나누어서 S_{i1}, S_{i2} ... S_{in}의 1 이상의 블록상태로 재구성하여 이들 각각의 블록상태들로 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩 초기상태로 설정하여 블록별로 병렬적으로 수행되어지는 것도 가능하다.

<50> 이와 같은 또 다른 3가지 사례들에서 1차 이후의 모델검사나 특성검사를 2이상의 정식검증 툴로 병렬적으로 수행하는 것이 유리하지만, 만일 사용가능한 정식검증 툴이 하나인 경우에는 2 이상의 모델검사나 특성검사들을 병렬적으로 수행하는 대신에 순차적으로 수행하는 것도 가능하다. 이와 같은 순차적인 수행 과정에서

는 (0, T_1), (T_1, T_2), ... (T_{n-1} T)으로 시간병렬가능정보가 구성된 경우에는 검증시간에서 뒤에 있는 (T_{n-1}, T)부터 (T_{n-1}, T_{n-2}) ... (T_1, T_2), 그리고 제일 앞에 있는 (0, T_1) 등의 순서를 정하여 진행하면서 진행 도중에 설계 오류의 위치와 원인이 밝혀지게 되면 이 이후의 정식검증 과정을 생략하는 것이 가능하다.

<51> 일곱 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 FPGA 보드로 구성되고, 상기 1 이상의 하드웨어에뮬레이터와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 하드웨어에뮬레이터로 하여 하드웨어에뮬레이터를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행한다. 이와 같은 방법의 장점은 수십억원 이상으로 매우 고가인 하드웨어에뮬레이터의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 하드웨어에뮬레이터의 이용률을 높일 수 있는 것은, 하드웨어에뮬레이터만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 하드웨어에뮬레이션을 진행하게 되면 에뮬레이션의 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 하드웨어에뮬레이터를 사용하여 진

행하면서 1차이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 에뮬레이션의 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 하드웨어에뮬레이터를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 하드웨어에뮬레이터의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 FPGA 보드를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 FPGA 보드를 이용하여 2 이상의 검증구간을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 FPGA 보드의 수행 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 하드웨어에뮬레이터를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 FPGA 보드의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 FPGA 보드를 이용한 검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(에뮬레이션 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. 또 다른 하나는 1차 검증 실행(에뮬레이션 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여

탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다.

<52> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1} T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 FPGA 보드를 통한 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 검증 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 에뮬레이션을 통한 검증 실행에서 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 검증 실행에서 2 이상의 FPGA 보드를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1} T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<53> 여덟 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 FPGA 보드로 구성되고, 상기 1 이상의 시뮬레이션가속기와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 시뮬레이션가속기로 하여 시뮬레이션가속기를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행한다. 이와 같은 방법의 장점은 수억원 이상으로 매우 고가인 시뮬레이션가속기의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 시뮬

레이션가속기의 이용률을 높일 수 있는 것은, 시뮬레이션가속기만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션가속을 진행하게 되면 시뮬레이션가속의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 시뮬레이션가속기를 사용하여 진행하면서 1차 이후의 검증실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 시뮬레이션가속의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 시뮬레이션가속기를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 고가의 시뮬레이션가속기의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 FPGA 보드를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을
사용한다. 한가지 사용할 수 있는 방법은 2 이상의 FPGA 보드를 이용하여 2 이상의 검증구간을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 FPGA 보드의 수행 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 시뮬레이션가속기를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 FPGA 보드의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 FPGA 보드를 이용한 검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소

한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(시뮬레이션가속 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. 또 다른 하나는 1차 검증 실행(시뮬레이션가속 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다.

<54> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1} T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 FPGA 보드를 통한 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 검증 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 시뮬레이션가속 실행을 통한 검증 실행의 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 검증 실행에서 2 이상의 FPGA 보드를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1} T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<55> 아홉 번째 일례로, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1

이상의 프로토타이핑시스템과 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 FPGA 보드로 구성되고, 상기 1 이상의 프로토타이핑시스템과 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 프로토타이핑시스템으로 하여 프로토타이핑시스템을 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행한다. 이와 같은 방법의 장점은 프로토타이핑시스템의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 프로토타이핑시스템의 이용률을 높일 수 있는 것은, 프로토타이핑시스템만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 프로토타이핑시스템을 실행시키게 되면 프로토타이핑시스템의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 프로토타이핑시스템을 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 프로토타이핑시스템의 실행 속도 저하를 최소화시키면서 동시에 1차 검증실행이 완료되는 즉시에 프로토타이핑시스템을 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 프로토타이핑시스템의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 FPGA 보드를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행하는 경우

에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 FPGA 보드를 이용하여 2 이상의 검증구간을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 FPGA 보드의 수행 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 프로토타이핑시스템을 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 FPGA 보드의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 FPGA 보드를 이용한 검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(프로토타이핑시스템 실행) 전체과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. 또 다른 하나는 1차 검증 실행(프로토타이핑시스템 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다.

<56> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1} T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 FPGA 보드를 통한 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록

들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 검증 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 프로토타이핑시스템의 실행을 통한 검증 실행의 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 검증 실행에서 2 이상의 FPGA 보드를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1} T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<57> 열 번째 일례로서, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 시뮬레이터와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 FPGA 보드로 구성되고, 상기 1 이상의 시뮬레이터와 1 이상의 컴퓨터는 컴퓨터네트워크로 연결되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 1 이상의 시뮬레이터로 하여 시뮬레이터를 사용하여 1차 검증 실행을 수행 하고(2이상의 시뮬레이터를 이용하는 경우에는 분산 시뮬레이션(distributed simulation)을 수행함), 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행한다. 이와 같은 방법의 장점은 1 이상의 시뮬레이터의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 시뮬레이터의 이용률을 높일 수 있는 것은, 시뮬레이션만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션을 진행하게 되면 시뮬레이션의 실행 속도가 그렇지 않

는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 시뮬레이션을 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 시뮬레이션의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 상기 1이상의 시뮬레이션을 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 시뮬레이터의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 FPGA 보드를 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 FPGA 보드를 사용하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 FPGA 보드를 이용하여 2 이상의 검증구간을 병렬적으로 진행하고, 필요시에는 이들 2 이상의 FPGA 보드의 수행 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 시뮬레이터를 이용하여 1차 검증 실행에서 과정에서 1차 이후의 2 이상의 FPGA 보드의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 FPGA 보드를 이용한 검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보에는 두가지가 있을 수 있다. 하나는 1차 검증 실행(시뮬레이션 실행)전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하면서 동시에 DUV에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 시간병렬가능정보이다. 또 다른 하나는 1차 검증

실행(시뮬레이션 실행) 전체 과정에서 DUV에 존재하는 2 이상들의 블록들 각각에 존재하는 모든 입력과 입출력들에 대하여서 검증 실행 전체 구간에 대하여 탐침을 수행하여 저장하여 얻어지는 공간병렬가능정보이다.

<58> 시간병렬가능정보는 1차 검증 실행 구간 (0, T)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1} T)로 나눌 수 있게 하고 이들 구간들을 각각 독립적으로 1차 이후의 FPGA 보드를 통한 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 공간병렬가능정보는 DUV 전체를 2 이상의 블록들 B_1, B_2, ... B_n으로 나눌 수 있게 하고, 이들 블록들을 각각 독립적으로 1차 이후의 검증 실행에서 2 이상의 FPGA 보드들 각각에서 병렬적으로 수행되어지는 것을 가능하게 한다. 또한 1차 시뮬레이션 실행을 통한 검증 실행의 과정에서 시간병렬가능정보와 공간병렬가능정보를 동시에 수집하여 1차 이후의 검증 실행에서 2이상의 FPGA 보드를 사용하여 (0, T_1), (T_1, T_2), ... (T_{n-1} T)와 B_1, B_2, ... B_n 모두에서 병렬적인 수행도 가능하다.

<59> 1차 이후의 검증 실행을 FPGA 보드를 이용하는 경우에 시간병렬가능정보를 이용하게 되려면 FPGA 보드상의 1 이상의 해당 FPGA에 구현된 DUV 전체 또는 DUV 일부분에 존재하는 모든 플립플롭과 래치 그리고 메모리에 대한 쓰기탐침과 경우에 따라서는 읽기탐침이 필요하다. 이와 같은 FPGA에 구현된 검증대상에 대하여 읽기탐침 내지는 쓰기탐침을 수행하는 방법은 미국특허 USUS6,701,491와 PCT 출원중인 특허(PCT/KR01/01092)에 자세히 나와 있음으로 생략하기로 한다. 또한 미국특허

USUS6,701,491와 PCT 출원중인 특허(PCT/KR01/01092)를 이용하면 FPGA보드에 구현된 DUV에 대해서도 100% 가시도를 갖도록 하는 것도 가능하다.

<60> 위에서 설명한 사례들에서 1차 이후의 시뮬레이션을 2 이상의 FPGA 보드를 이용하여 병렬적으로 수행하는 것이 유리하지만, 만일 사용가능한 FPGA 보드가 하나인 경우에는 2 이상의 구간별 혹은 블록별 설계 검증들을 병렬적으로 수행하는 대신에 순차적으로 수행하는 것도 가능하다. 이와 같은 구간별 순차적인 수행 과정에서는 $(0, T_1)$, (T_1, T_2), ... $(T_{n-1} T)$으로 시간병렬가능정보가 구성된 경우에는 검증시간에서 뒤에 있는 (T_{n-1}, T)부터 (T_{n-1}, T_{n-2}) ... (T_1, T_2), 그리고 제일 앞에 있는 $(0, T_1)$ 등의 순서를 정하여 진행하면서 진행 도중에 설계 오류의 위치와 원인이 밝혀지게 되면 이 이후의 FPGA 보드를 이용한 설계 검증을 생략하는 것이 가능하다. 뿐만 아니라 1 이상의 FPGA 보드에 장착된 1 이상의 FPGA 보드에서의 로직용량이 DUV의 게이트 용량보다 작은 경우에는 DUV내에 존재하는 블록별이나 DUV를 임의로 분할(partition)하여 생성되는 블럭별로 상기 1 이상의 FPGA에 구현하고 검증 실행을 순차적으로 진행한다.

<61> 마지막 일례로서, 본 발명에서 제안하는 검증 장치는 검증 소프트웨어와 1 이상의 시뮬레이터와 1 이상의 컴퓨터와 이 1 이상의 컴퓨터에 인스톨된 1 이상의 정식검증 툴(모델검사기 내지는 특성검사기)로 구성되어 있다. 이와 같은 검증 장치에서의 검증 수행은 1차 검증 실행 플랫폼을 1 이상의 시뮬레이터로 하여 시뮬레이터를 사용하여 1차 검증 실행을 수행하고, 1차 이후의 검증 실행을 1 이상의 정

식검증 툴을 사용하여 진행한다. 이와 같은 방법의 장점은 시뮬레이터의 이용률을 높일 수 있을 뿐만 아니라, 검증의 실행 속도도 높일 수 있다. 우선 시뮬레이터의 이용률을 높일 수 있는 것은, 시뮬레이션만으로 설계 검증을 수행하는 경우에는 설계 오류를 발견하고 수정하기 위하여 설계 코드에 존재하는 모든 시그널들과 변수들을 탐침대상으로 선정하여 수행하게 된다. 이와 같이 설계 코드에 대하여 100% 가시도를 가질 수 있도록 지정하고 시뮬레이션을 진행하게 되면 시뮬레이션의 실행 속도가 그렇지 않는 경우와 비교하여서 대략 2배 이상 또는 그 이상 떨어지게 된다. 따라서 이와 같은 기존의 방법 대신에 본 발명에서의 방법은 1차 검증 실행을 1 이상의 시뮬레이터를 사용하여 진행하면서 1차 이후의 검증 실행에 필요한 최소한의 시그널들과 변수들만을 검증 실행 과정에서 탐침하여 탐침에 의한 시뮬레이션의 실행 속도 저하를 최소화시키면서 동시에 1차 검증 실행이 완료되는 즉시에 상기 1 이상의 시뮬레이터를 동일 설계에 대한 다른 검증이나 다른 설계의 검증 작업에 사용될 수 있게 함으로서 시뮬레이터의 이용률을 높일 수 있게 한다. 1차 이후의 검증 실행은 1 이상의 정식검증 툴을 사용하여 진행된다. 1차 이후의 검증 실행을 1 이상의 정식검증 툴로 사용하여 모델 검사나 특성 검사 과정을 통하여 진행하는 경우에서 검증 실행 속도를 높일 수 있는 방법들로는 다음과 같은 방법들을 사용한다. 한가지 사용할 수 있는 방법은 2 이상의 정식검증 툴을 이용하여 2 이상의 모델 검사나 특성 검사를 병렬적으로 진행하고, 필요시에는 이들 2 이상의 모델 검사나 특성 검사의 결과를 통합하여 전체 검증 실행의 결과로 이용할 수 있다. 이를 위해서 1 이상의 시뮬레이터를 이용하여 1차 검증 실행에서 과정에서 1차 이후

의 2 이상의 정식검증의 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보를 수집하게 된다. 이와 같은 시뮬레이션 실행을 병렬적으로 수행될 수 있는데 필요한 최소한의 정보는 DUV에 존재하는 1 이상의 상태정보이다. 이는 1차 검증 실행(시뮬레이션 실행) 전체 과정에서 일정 간격(예로 검증 시간으로 50,000 나노초마다 한번씩)으로나 사용자가 원하는 1 이상의 시점에서 DUV에 존재하는 상태정보를 저장하여 얻어지는 시간병렬가능정보이다.

<62> 시간병렬가능정보는 1차 검증 실행 구간 (0, T_n)를 2 이상의 구간들 (0, T_1), (T_1, T_2), ... (T_{n-1} T_n)로 나누어서 0, T_1, T_2, ... T_{n-1} T_n 시점들 각각에서의 상태들 S_0, S_1, S_2, ... S_{n-1}, S_n 들을 모두 이용하거나 또는 이들 중에서 특정 상태들을 선택적으로 이용하여 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩으로 초기상태로 설정하여 병렬적으로 수행되어지는 것이 가능하다. 뿐만 아니라 DUV가 B_1, B_2, ... B_n과 같이 1이상의 블록들로 나누어져서 구성되어 있는 경우에 상기 특정 검증시점에서의 DUV의 특정 상태 S_i를 이들 블록별로 나누어서 S_{i1}, S_{i2}, ... S_{in}의 1 이상의 블록상태로 재구성하여 이들 각각의 블록상태들로 모델검사나 특성검사를 2 이상의 정식검증 툴들 각각에 하나씩 초기상태로 설정하여 블록별로 병렬적으로 수행되어지는 것도 가능하다.

<63> 만일 1차 검증 실행이 시뮬레이터를 검증 플랫폼으로 하는 시뮬레이션이라면 시간병렬가능정보에서 필요한 1 이상의 검증사이클의 특정시점에서 상태정보를 얻

기 위하여 설계 코드에 추가되는 부가 코드가 직접 상태정보를 추출하는 것도 가능하고(상세한 것은 미국특허 USUS6,701,491와 PCT 출원중인 특허 PCT/KR01/01092를 참조함), HDL 시뮬레이터의 save 명령어를 사용하여 1차적으로는 시뮬레이션상태를 우선 저장하고 이로부터 후에 상태정보를 추출하는 것도 가능하고, 또는 HDL 시뮬레이터에 상태정보 저장을 가능하게 하는 명령어가 존재한다면 이를 집적 이용하는 것도 가능하다.

<64> 상기 목적 외에 본 발명의 다른 목적 및 이점들은 첨부한 도면을 참조한 실시 예에 대한 상세한 설명을 통하여 명백하게 드러나게 될 것이다.

<65> 도1 (a)는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예를 개략적으로 도시한 도면이다.

<66> 도1 (b), (c) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예들을 개략적으로 도시한 도면이다.

<67> 도1 (d) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 하드웨어에뮬레이터로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<68> 도1 (e) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이

상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 시뮬레이션가속기로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<69> 도1 (f) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면이다.

<70> 도1 (g) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 시뮬레이션가속기로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<71> 도1 (h) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 하드웨어에뮬레이터로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<72> 도1 (i) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼

을 프로토타이핑시스템으로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<73> 도1 (j) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 프로토타이핑시스템으로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<74> 도1 (k) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 하드웨어에뮬레이터로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<75> 도1 (l) 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 시뮬레이션가속기로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<76> 도1 (m) 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른

일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 프로토타이핑시스템으로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 시뮬레이터로 구성한 일 예를 개략적으로 도시한 도면이다.

<77> 도1 (n) 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 하드웨어에뮬레이터로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<78> 도1 (o) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 시뮬레이션가속기로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<79> 도1 (p) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 프로토타이핑시스템으로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 하나의 모델검사기 내지는 특성검사기로 구성한 일 예를 개략적으로 도시한 도면이다.

<80> 도1 (q) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른

일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템으로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 1 이상의 FPGA 보드로 구성한 일 예를 개략적으로 도시한 도면이다.

<81> 도1 (r) 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 1 이상의 시뮬레이터로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 FPGA 보드로 구성한 일 예를 개략적으로 도시한 도면이다.

<82> 도1 (s) 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 또 다른 일 예를 개략적으로 도시한 도면으로, 구체적으로는 1차 검증 실행을 위한 플랫폼을 1 이상의 모델검사기 내지는 특성검사기로 구성하고, 1차 이후의 검증 실행을 위한 플랫폼은 FPGA 보드로 구성한 일 예를 개략적으로 도시한 도면이다.

<83> 도2 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 시뮬레이터들로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<84> 도3 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의

검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 모델검사기 내지는 특성검사기들로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<85> 도4 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 시뮬레이터와 모델검사기 내지는 특성검사기를 같이 사용하여 병렬적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<86> 도5 는, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 하나의 시뮬레이터로 순서적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<87> 도6 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 하나의 모델검사기 내지는 특성검사기로 순서적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<88> 도7 은, 컴퓨터에서 운영되는 본 발명의 검증 소프트웨어와 상이한 2 이상의 검증 플랫폼들과 컴퓨터로 구성된 본 발명에 관한 설계 검증 장치의 일 예에서 1차 검증 실행을 하드웨어에뮬레이터 내지는 시뮬레이션가속기 내지는 프로토타이핑시스템에서 수행하고, 1차 이후의 검증 실행은 2 이상의 FPGA 보드들로 병렬적으로 수행하는 과정을 개략적으로 도시한 도면이다.

<89> 도8 은 본 발명에서의 검증 장치를 이용하여 설계에 존재하는 1 이상의 설계 오류를 발견하여 수정하는 과정을 개략적으로 도시한 순서도이다.

<90> 도9 는 본 발명에서의 검증 장치를 이용하여 설계 검증을 수행하는 과정을 개략적으로 도시한 순서도이다.

【발명의 효과】

<91> 상술한 바와 같이, 본 발명에 따른 검증 장치 및 이를 이용한 설계 검증 방법의 목적은 초대규모급 설계 검증을 위하여 검증 실행을 상이한 검증 플랫폼들인 시뮬레이터, 시뮬레이션가속기, 하드웨어에뮬레이터, 프로토타이핑시스템, 정식검증툴들을 혼용하여 수행하는 과정을, 1차 특정 검증 플랫폼을 이용한 검증 실행을 통해서 1차 이후에 추가적으로 진행될 검증 실행들을 매우 빠르고 효과적으로 진행하는 것에 필요한 최소한의 정보들을 수집하고, 이 정보를 이용하여 1차 실행 이후의 검증 실행은 상기 1차 특정 검증플랫폼과는 다른 플랫폼을 이용하여 병렬적으로 또는 순차적으로 진행함으로서 검증의 성능과 검증 플랫폼의 이용률과 검증의 효율성을 높이고 신속한 디버깅을 가능하게 한다.

<92> 이상 설명한 내용을 통해 당업자라면 본 발명의 기술사상을 일탈하지 아니하는 범위에서 다양한 변경 및 수정이 가능함을 알 수 있을 것이다. 따라서, 본 발명의 기술적 범위는 실시예에 기재된 내용으로 한정되는 것이 아니라 특허 청구의 범위에 의하여 정하여져야만 한다.

【특허청구범위】

【청구항 1】

검증 소프트웨어와 2 이상의 다른 검증 플랫폼을 구비하는 설계검증 장치에 있어서

상기 검증 소프트웨어는 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 자동화된 방식을 통하여 부가 코드나 부가회로를 부가하여 상기 2 이상의 다른 검증 플랫폼들 중의 1 이상의 특정 검증 플랫폼 상에서 1차 검증 실행을 수행하면서 1차 검증 실행 이후의 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 다른 검증 플랫폼 상에서 수행되는 검증 실행을 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 블록들에 대하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할 수 있도록 하고, 상기 1 이상의 1차 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 상기 최소한의 정보를 수집하고, 이 수집된 정보를 이용하여 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 1차 이후의 1회 이상의 검증 실행을 신속하게 수행하는 것을 가능하게 하는 설계 검증 장치.

【청구항 2】

제 1 항에 있어서,

상기 2 이상의 다른 검증 플랫폼들 중의 특정 검증 플랫폼 상에서 1차 검증 실행을 수행하면서 1차 검증 실행 이후의 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 수행되는 검증 실행의 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 블록들에 대하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할 수 있도록 하고, 상기 1차 검증 실행을 위한 1 이상의 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 상기 최소한의 정보를 수집하고, 이 수집된 정보를 이용함으로서 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 1차 이후의 1회 이상의 검증 실행을 0 검증 사이클 시간에서부터 시작되지 않아도 되도록 함으로서 검증을 신속하게 수행하는 것을 가능하게 하는 설계 검증 장치.

【청구항 3】

제 1 항에 있어서,

2 이상의 다른 검증 플랫폼들 중의 특정 1 이상의 검증 플랫폼 상에서 1차 검증 실행을 수행하면서 1차 검증 실행 이후의 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 수행되는 검증 실행의 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 블록들에 대하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할

수 있도록 하고, 상기 1차 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 상기 최소한의 정보를 수집하고, 이 수집된 정보를 이용함으로서 상기 1차 검증 실행을 위한 1 이상의 검증 플랫폼을 이용한 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 검증 플랫폼 상에서 1차 이후의 1회 이상의 검증 실행에서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 1 이상의 특정 블록들에 대해서만 검증 실행을 할 수 있도록 함으로서 검증을 신속하게 수행하는 것을 가능하게 하는 설계 검증 장치.

【청구항 4】

2 이상의 검증 플랫폼을 혼합적으로 이용하여 설계 검증 실행을 수행함으로서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 설계 오류의 위치를 알아내고 이를 수정하는 설계 검증 방법에 있어서,

상기 설계 검증 실행을 상기 2 이상의 검증 플랫폼 중의 1 이상의 특정 검증 플랫폼 상에서의 1차 검증 실행과 1차 이후의 검증 실행은 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서의 1회 이상의 검증 실행들로 나누어서 수행하며, 상기 1차 검증 실행을 수행하면서 1차 검증 실행 이후의 다른 검증 플랫폼 상에서 수행되는 1회 이상의 검증실행의 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 블록들에 대하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할 수 있도록 하고, 상기 1 이상의 1차 검증 플랫폼을 이용한 1차 검증 실행을 수행하면서 상기 최

소한의 정보를 수집하고, 이 수집된 정보를 이용함으로서 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 1차 이후의 1회 이상의 검증 실행에서 검증 실행의 특정 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 1 이상의 특정 블록들에 대해서만 검증 실행을 할 수 있도록 함으로서 검증을 신속하게 수행하는 것을 가능하게 검증 방법

【청구항 5】

제 4 항에 있어서,

2 이상의 검증 플랫폼을 혼합적으로 이용하여 설계 검증 실행을 수행함으로서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 설계 오류의 위치를 알아내고 이를 수정하는 설계 검증 방법에 있어서,

상기 설계 검증 실행을 상기 2 이상의 검증 플랫폼 중의 1 이상의 특정 검증 플랫폼 상에서의 1차 검증 실행과 1차 이후의 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서의 1회 이상의 검증 실행들로 나누어서 수행하며, 상기 1차 검증 실행을 수행하면서 1차 검증 실행 이후의 다른 검증 플랫폼 상에서 수행되는 1회 이상의 검증 실행에서 검증 실행의 특정 검증사이클 시간구간들이나 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 1 이상의 특정 블록들에 대하여 검증 실행들이 이루어질 수 있도록 하는데 필요한 최소한의 정보를 1차 검증 실행 과정에서 자동적으로 수집할 수 있도록 하고, 상기 1 이상의 1차 검증 플랫폼을 이용한 1차 검증 실

행을 수행하면서 상기 최소한의 정보를 수집하고, 이 수집된 정보를 이용하여 상기 1차 이후의 검증 실행을 상기 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나 이상에서 다른 1 이상의 검증 플랫폼 상에서 0 검증사이클 시간에서부터 시작되지 않아도 되도록 함으로서 검증을 신속하게 수행하는 것을 가능하게 하는 검증 방법

【청구항 6】

제 4 항 내지는 제 5 항에 있어서,

1차 이후의 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나이상에서 다른 1 이상의 검증 플랫폼 상에서의 1회 이상의 검증 실행 각각을 2 이상의 검증 플랫폼들 상에서 병렬적으로 수행함으로서 검증을 신속하게 수행하는 것을 가능하게 하는 검증 방법

【청구항 7】

제 6 항에 있어서,

1차 이후의 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 최소한 하나이상에서 다른 1 이상의 검증 플랫폼 상에서의 1회 이상의 검증 실행 각각을 수행하기 위한, 상기 2 이상의 검증 플랫폼들 구성을 둘 이상의 서로 상이한 각각 다른 검증 플랫폼들로 구성하여 이들을 병렬적으로 수행함으로서 검증을 신속하게 수행하는 것을 가능하게 하는 검증 방법

【청구항 8】

제 4 항 내지는 제 5 항에 있어서,

1차 이후의 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 다른 검증 플랫폼 상에서의 1회 이상의 검증 실행 각각을 하나의 검증 플랫폼 상에서 순서적으로 수행함으로서 검증을 수행하는 것을 가능하게 하는 검증 방법

【청구항 9】

제 4 항 내지는 제 5 항 내지는 제 6 항 내지는 제 7 항 내지는 제 8 항에 있어서,

1차 이후의 1 이상의 검증 실행에 있어서, 검증 실행 각각에서의 실행하고자 하는 검증사이클 전체구간이나 검증사이클 특정구간을 1 이상의 특정 검증 플랫폼을 이용한 1차 검증 실행에서 수집된 2 이상의 상태정보들에서부터 시작될 수 있도록 1차 검증 실행에 사용된 1 이상의 특정 검증 플랫폼과는 다른 하나의 검증 플랫폼을 2회 이상 순서적으로 설정하고 2회 이상 순서적으로 상기 다른 하나의 검증 플랫폼을 실행시키는 과정에 있어서, 1차 검증 실행 시에 상태정보를 저장하는 시점이 검증시간 적으로 제일 뒤에 있는 상태정보를 우선 상기 하나의 검증 플랫폼에 설정하여 이 상태정보를 가지고 진행하고, 이 후의 실행들도 상기 하나의 검증 플랫폼으로써 수행하는 과정에서 1차 검증실행 과정에서 상태정보를 저장한 시점이 검증시간 적으로 뒤에 있는 것들을 우선으로 하여 상기 하나의 검증 플랫폼에 설정하여 1차 이후의 검증 실행들 각각이 2 이상의 상태정보를 이용하여 검증시간 적으

로 제일 뒤에서부터 앞서는 순서대로 진행하면서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 시그널들과 변수들에 대한 탐침을 수행하는 설계 검증 방법

【청구항 10】

제 1 항 내지는 제 2 항 내지는 제 3 항 내지는 제 4 항 내지는 제 5 항에 있어서,

최소한의 정보 수집이 1차 검증 실행 과정에서 일정 간격으로 혹은 특정 시점들에서 DUV의 상태정보와 검증사이클 전체에서 DUV의 모든 입력과 입출력 정보를 1 이상의 파일 형태로 저장하는 포함하는 설계 검증 방법

【청구항 11】

제 1 항 내지는 제 2 항 내지는 제 3 항 내지는 제 4 항 내지는 제 5 항에 있어서,

1차 검증 실행에 사용된 검증 플랫폼이 1 이상의 시뮬레이터인 경우에 있어서, 최소한의 정보 수집이 1차 검증 실행 과정에서 일정 간격으로 혹은 특정 시점들에서 시뮬레이션상태를 1 이상의 파일 형태로 저장하는 것을 포함하는 설계 검증 방법

【청구항 12】

제 11 항에 있어서,

상기 1 이상의 시뮬레이터를 이용한 1차 검증 실행 시행 중에 1 이상의 시점

에서 시뮬레이션 상태를 저장하는 방법을 시뮬레이터의 save 명령어를 사용하고, 1 차 이후의 검증 실행들을 상기 선정된 1 이상의 시뮬레이션 상태에서부터 추출된 상태정보에서부터 시작될 수 있도록 하는 설계 검증 방법

【청구항 13】

제 1 항 내지는 제 2 항 내지는 제 3 항 내지는 제 4 항 내지는 제 5 항에 있어서,

최소한의 정보 수집이 1차 검증 실행 과정에서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 특정 블록들의 입력과 입출력을 검증사이클 전체에서 지속적으로 탐침하여 1 이상의 파일 형태로 저장하는 것을 포함하는 설계 검증 방법.

【청구항 14】

제 6 항 내지는 제 7 항에 있어서,

1차 이후의 검증 실행 각각을 1차 검증 실행 과정에서 저장된 2 이상의 DUV 상태정보들 각각을 이용하여 2회 이상의 부분 검증 실행을 하는 것으로 전환하여, 컴퓨터 네트워크로 연결된 2 이상의 컴퓨터에 인스톨된 2 이상의 검증 플랫폼들을 이용하여 상기 2 이상의 상태정보들 각각을 상기 2 이상의 검증 플랫폼 각각에 설정하여 2회 이상의 부분 검증 실행이 서로 독립적으로 상기 2 이상의 검증 플랫폼으로써 병렬적으로 실행되게 하고, 실행된 2 이상의 결과들을 컴퓨터 네트워크를 이용하여 한 컴퓨터에 전송하고 통합하여 전체의 검증 결과로 제공하는 설계 검증

방법

【청구항 15】

제 6 항 내지는 제 7 항에 있어서,

1차 이후의 검증 실행 각각을 1차 검증 실행을 통하여 얻어진 2 이상의 블록들과 이들 블록들 각각의 모든 입력과 입출력을 1차 검증 실행 전과정에서 탐침한 2 이상의 탐침 파일들을 컴파일하여 얻어진 2 이상의 실행 파일들을 컴퓨터 네트워크로 연결된 2 이상의 컴퓨터에 인스톨된 2 이상의 검증 플랫폼들을 이용하여 독립적으로 병렬적으로 실행하고, 실행된 2 이상의 결과들을 컴퓨터 네트워크를 이용하여 한 컴퓨터에 전송하고 통합하여 전체의 검증 결과로 제공하는 설계 검증 방법

【청구항 16】

제 3 항에 있어서,

1차 검증 실행 시행 중에 1 이상의 시점에서 저장된 1 이상의 상태정보들 중에서 1 이상의 상태정보를 선정하여 1차 이후의 설계 검증들을 상기 선정된 1 이상의 상태정보에서부터 시작될 수 있도록 1차 검증 실행에서 사용된 검증 플랫폼과는 다른 검증 플랫폼을 1회 이상 설정한 후에 검증 실행을 1회 이상 진행하면서 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 시그널들과 변수들에 대한 탐침을 수행하는 설계 검증 방법

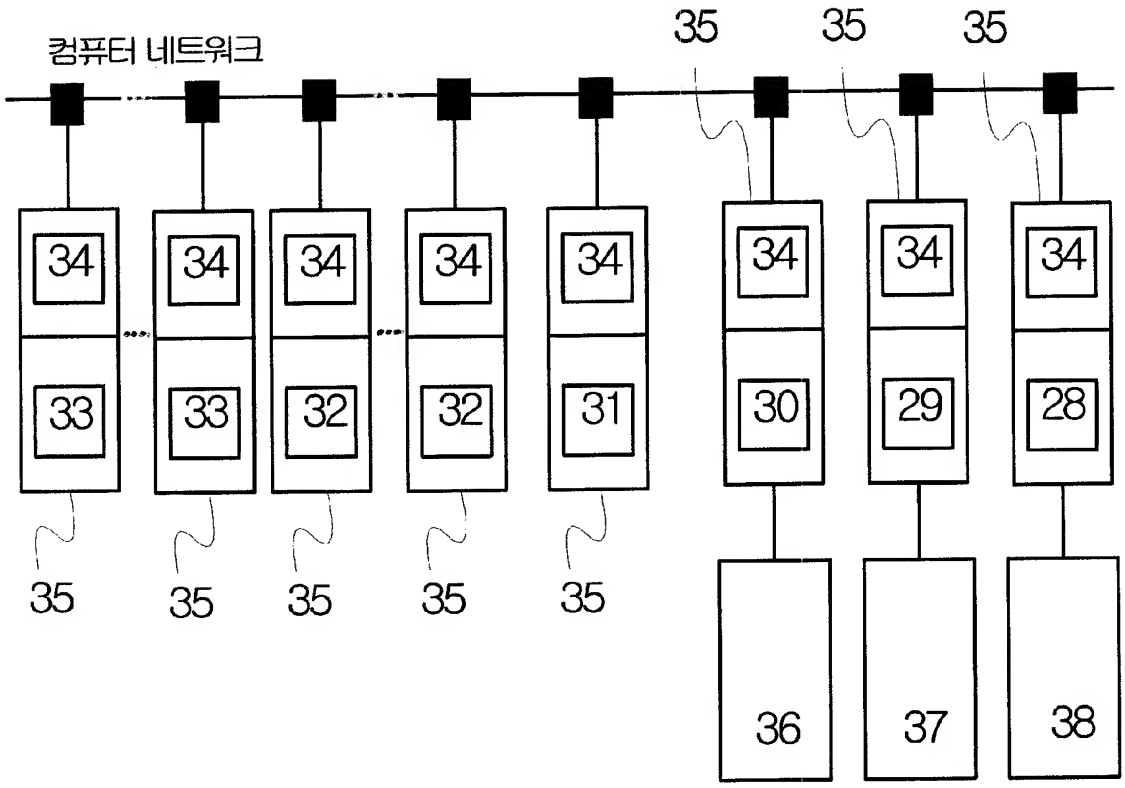
【청구항 17】

제 4 항에 있어서,

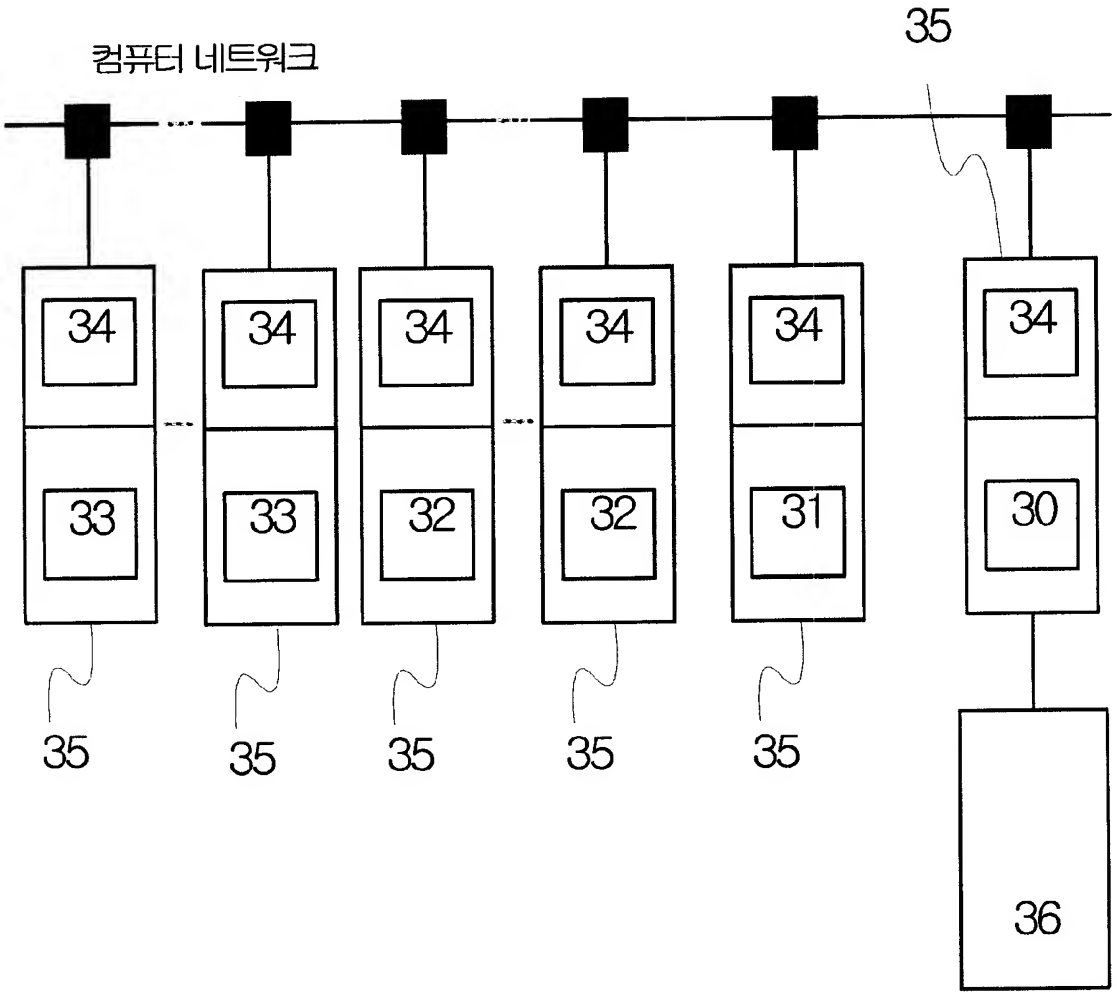
1차 검증 실행 중에 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 1 이상의 블록들에 대한 입력과 입출력들을 검증 실행의 검증사이클 전 과정에서 탐침하여 저장한 1 이상의 탐침 파일들 중에서 1 이상을 선정하여 1차 이후의 검증 실행들을 상기 선정된 1 이상의 탐침 파일과 해당 1 이상의 블록들을 이용하여 1차 검증 실행에서 사용된 검증 플랫폼과는 다른 검증 플랫폼을 위하여 컴파일하여 생성된 1 이상의 파일들 중에서 추가적인 탐침이 필요한 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 존재하는 시그널들과 변수들을 가지고 있는 해당 1 이상의 블록들에 관계되는 파일을 이용하여 상기 추가적인 탐침을 수행하는 설계 검증 방법

【도면】

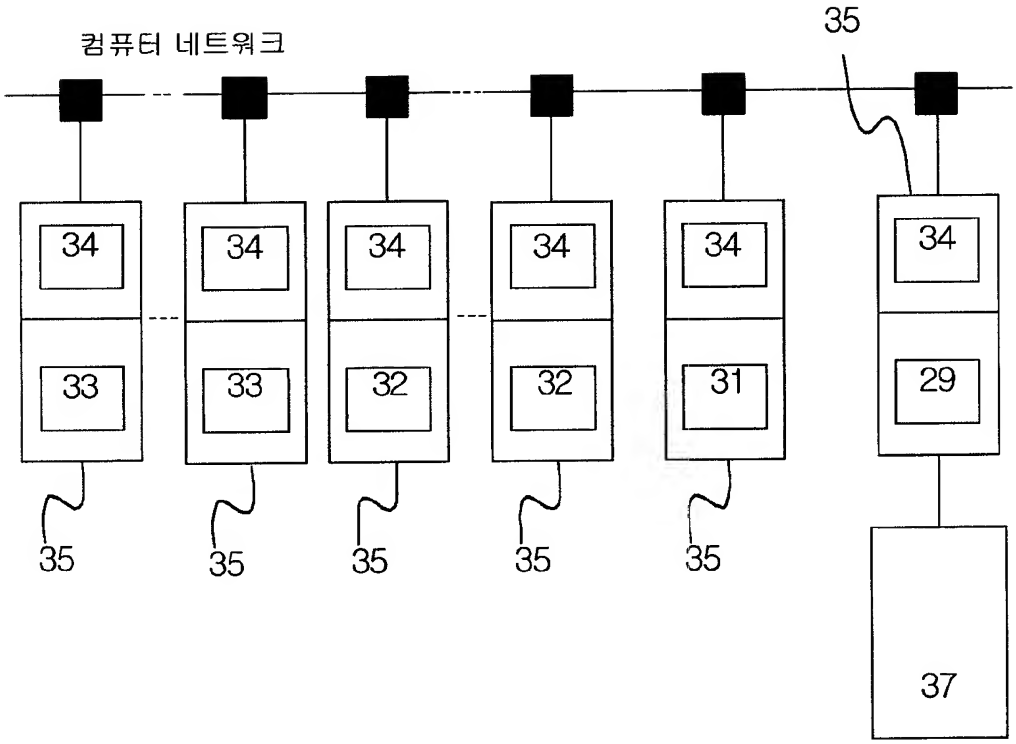
【도 1a】


컴퓨터 네트워크
35
35
35
34
34
34
34
34
34
34
34
34
33
33
32
32
31
30
29
28
35
35
35
35
35
36
37
38

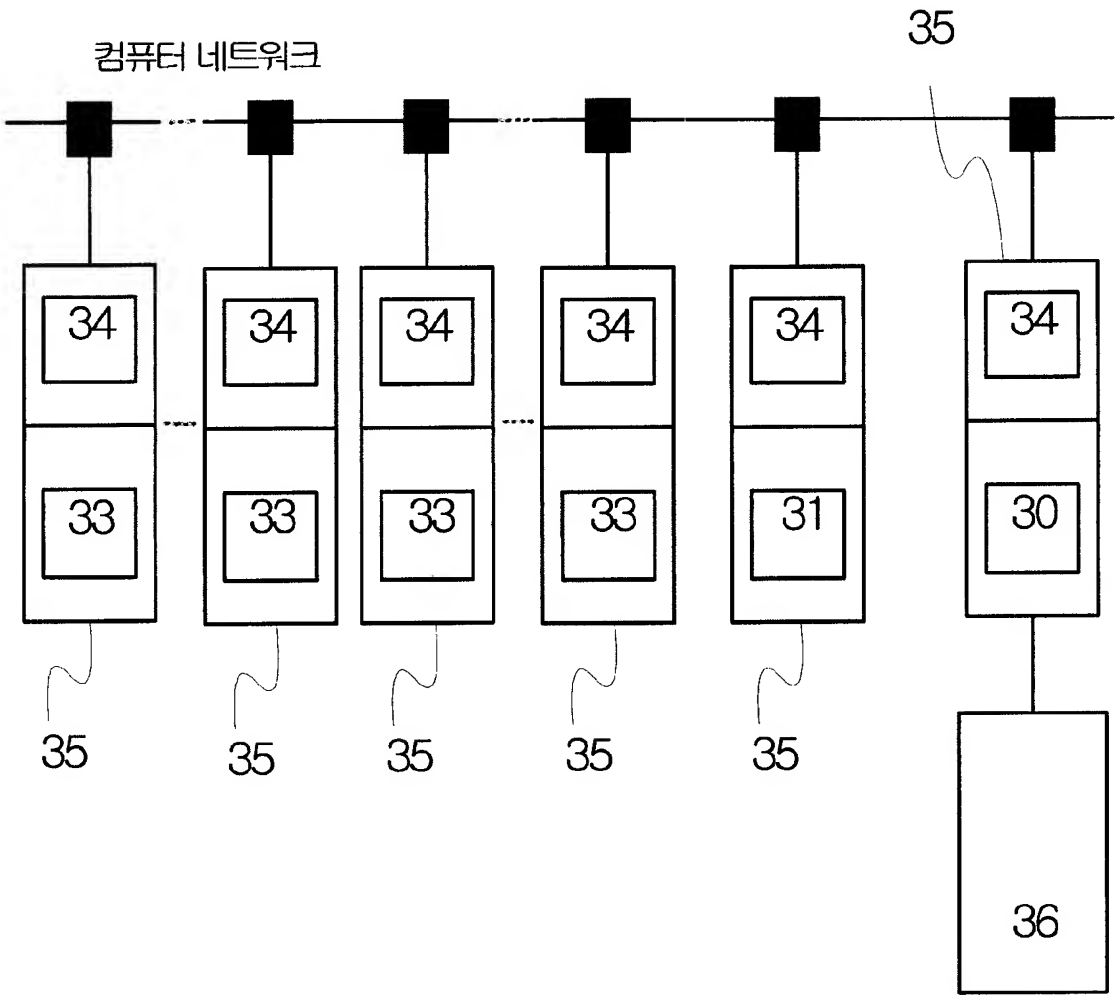
【도 1b】



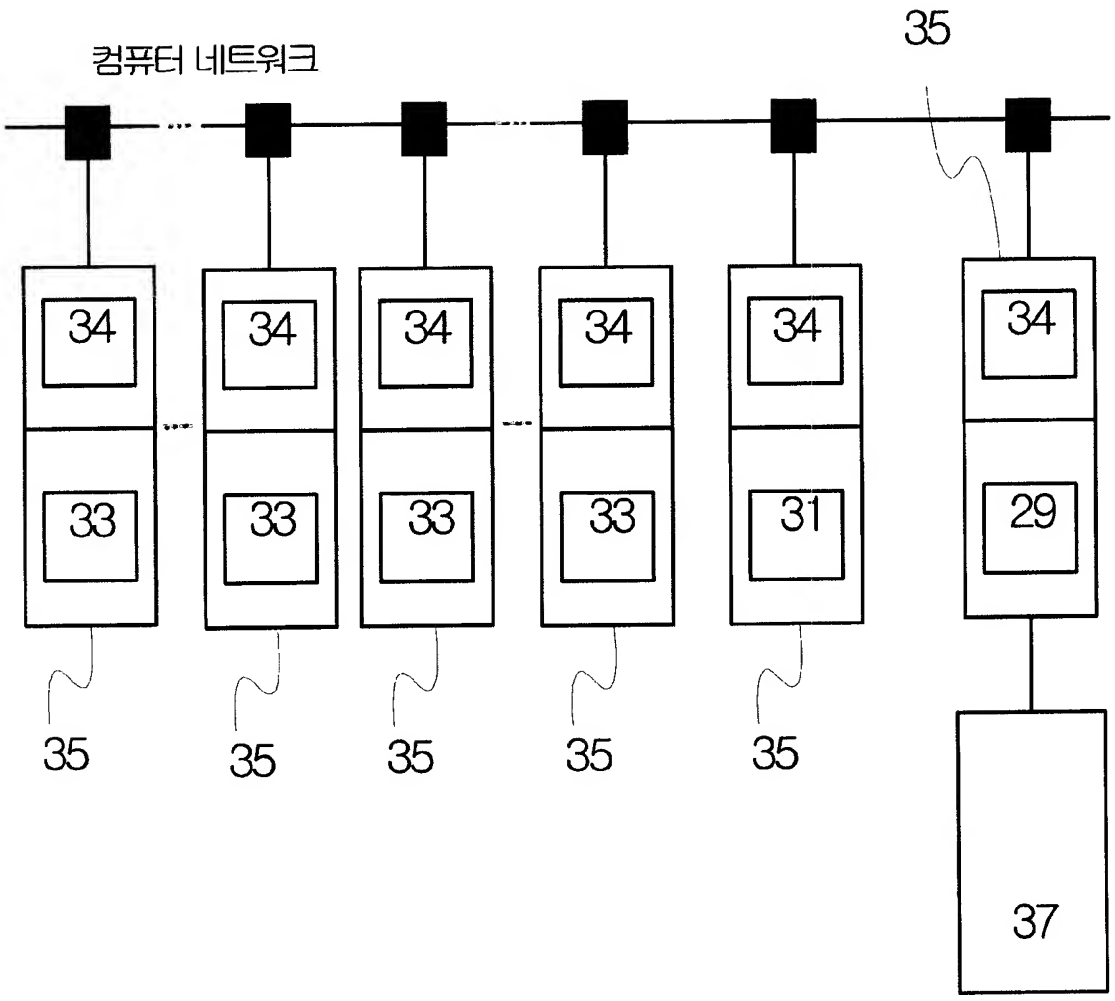
【도 1c】



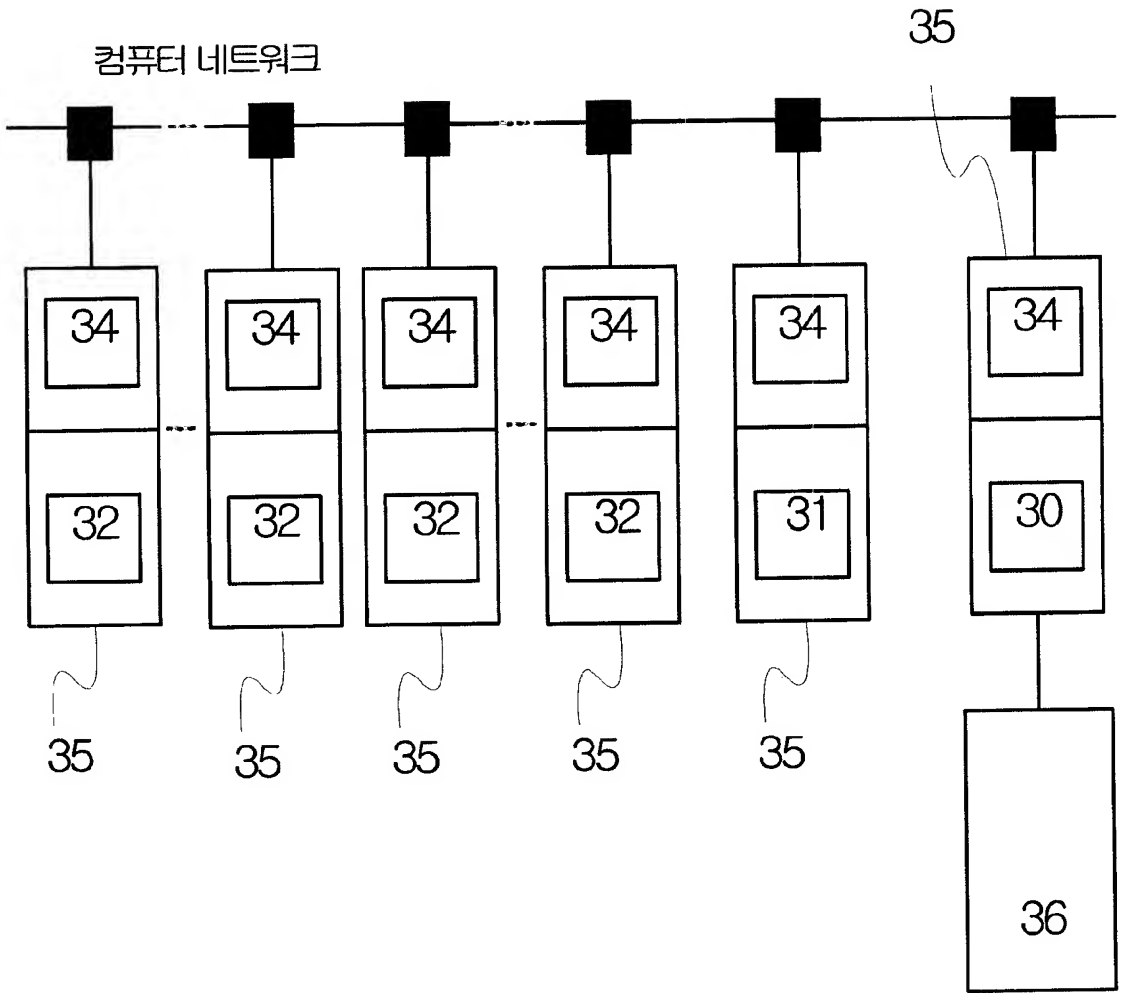
【도 1d】



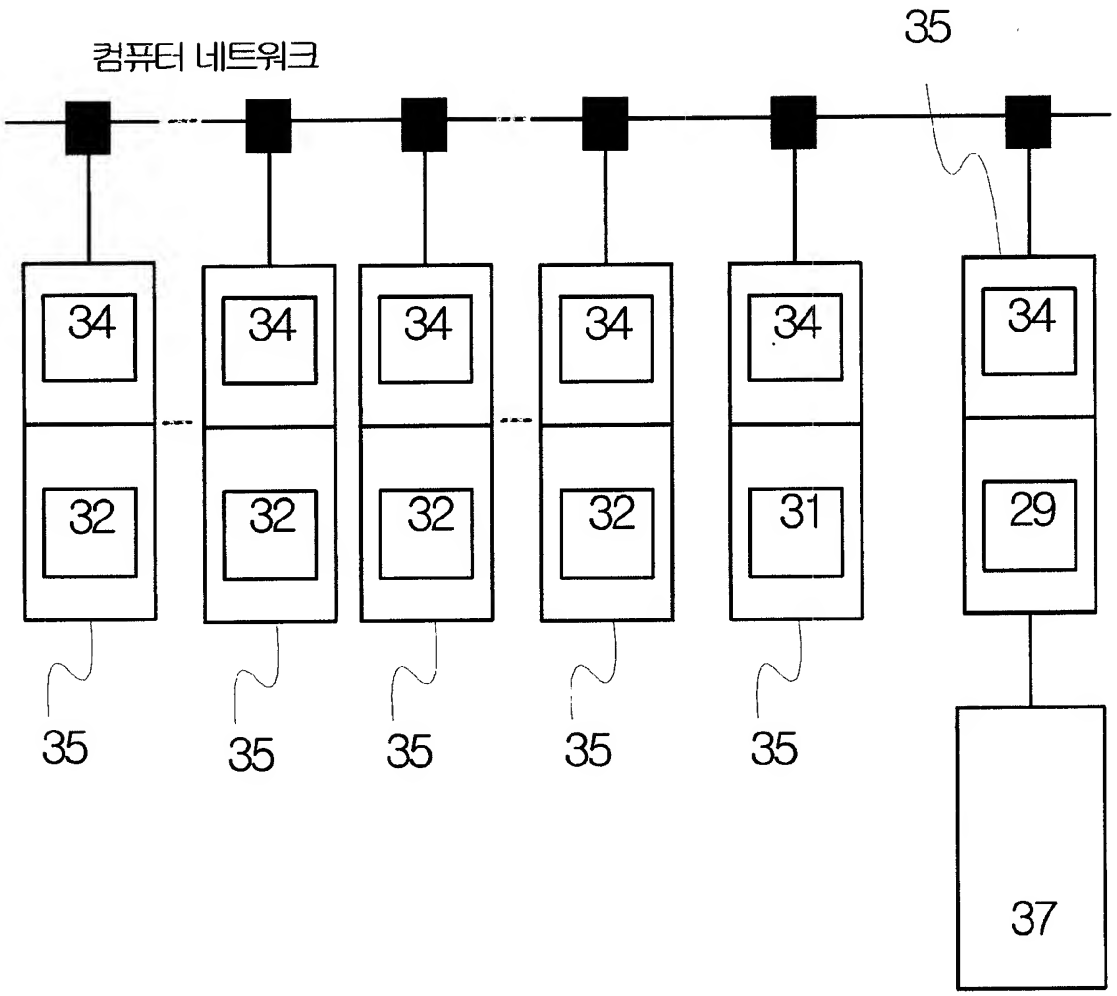
【도 1e】



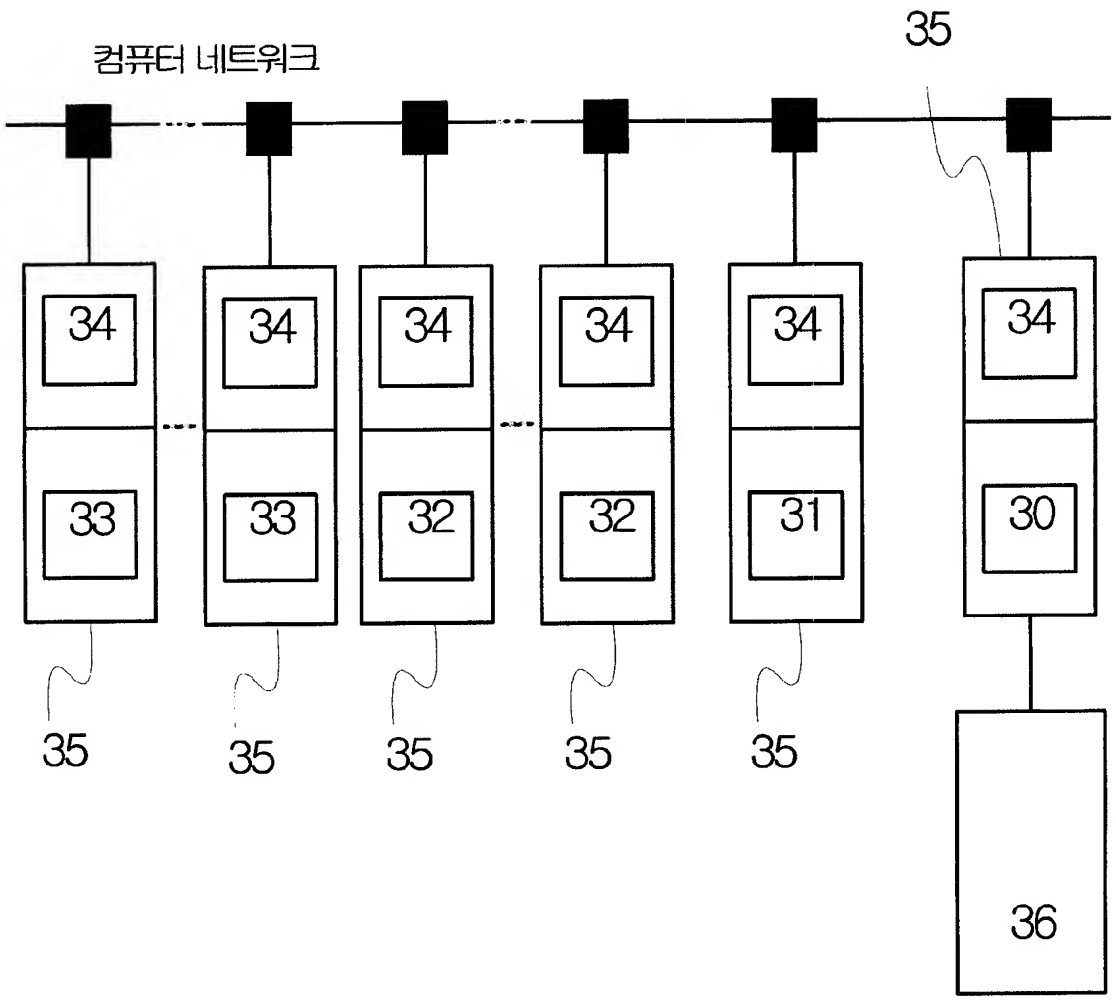
【도 1f】



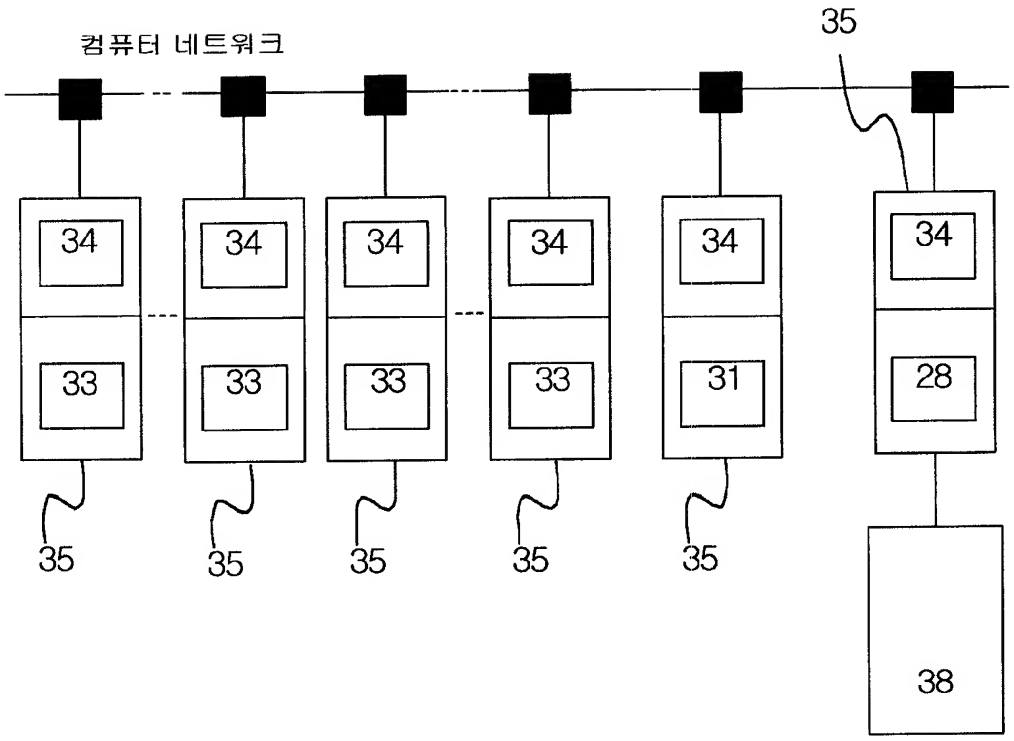
【도 1g】



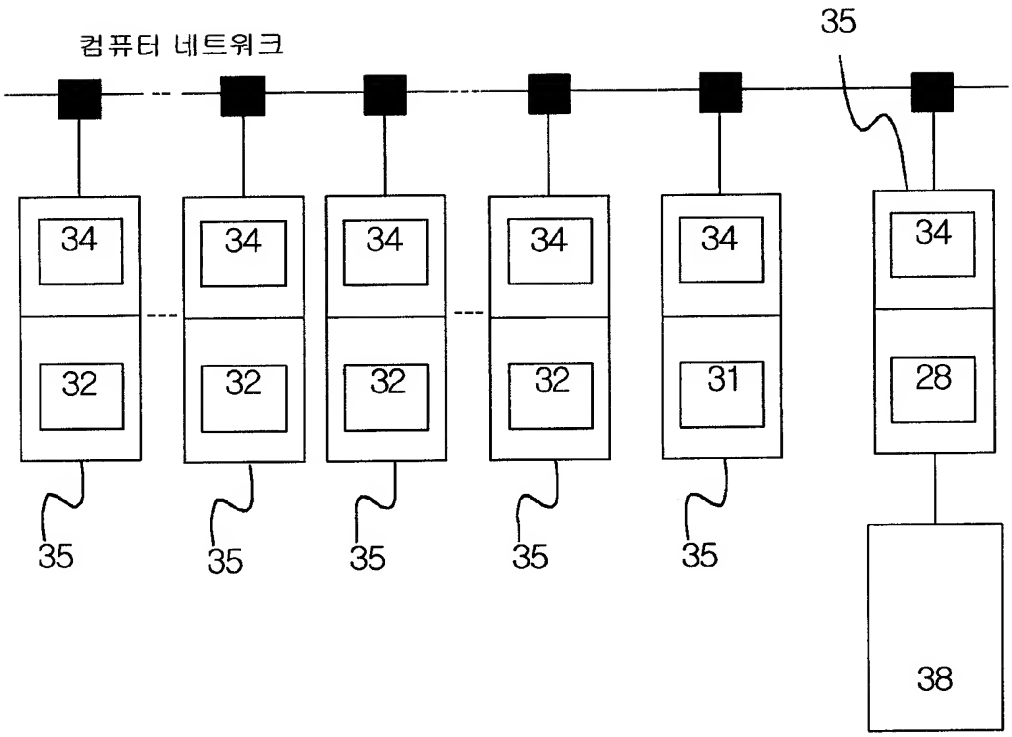
【도 1h】



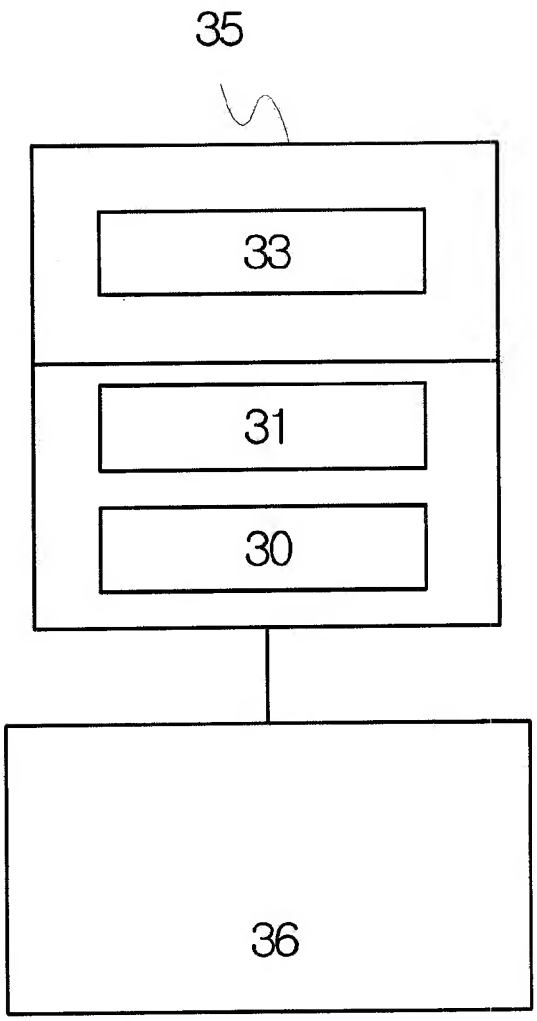
【도 1i】



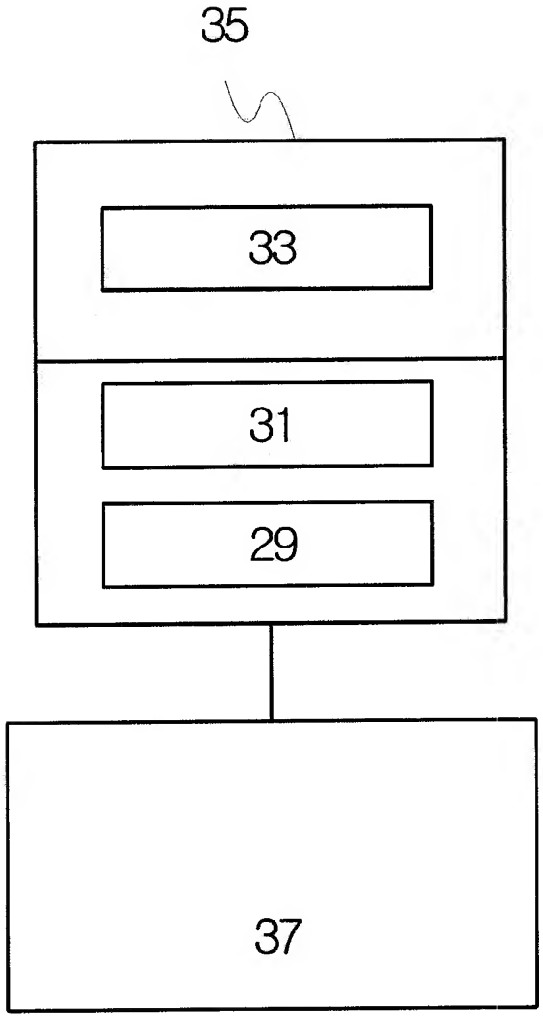
【도 1j】



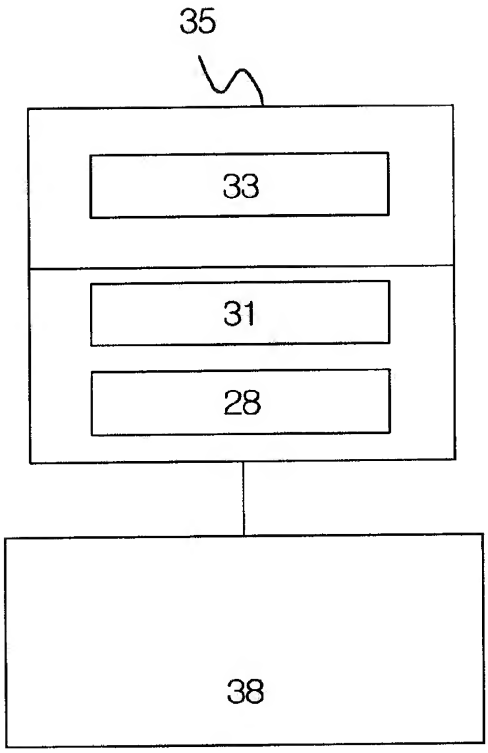
【도 1k】



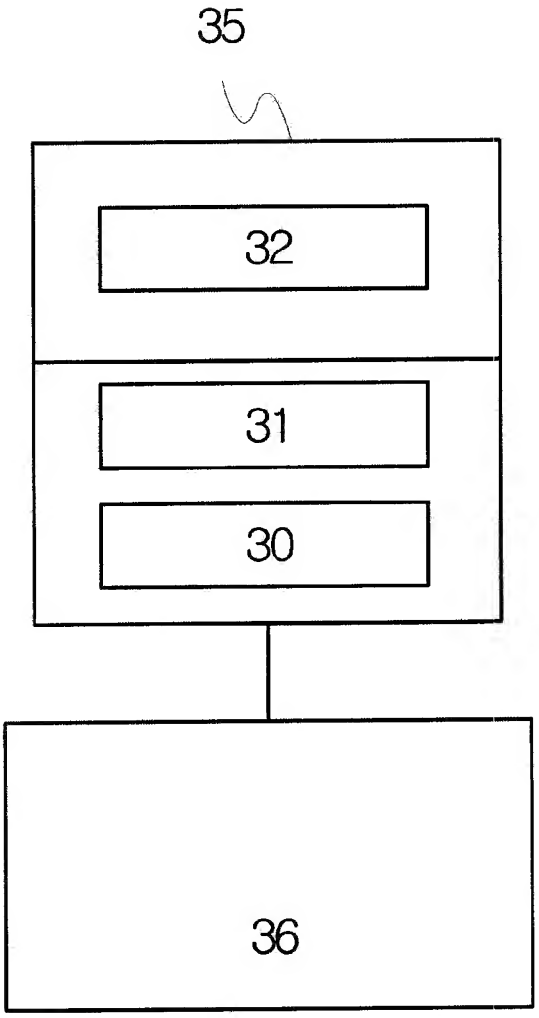
【도 11】



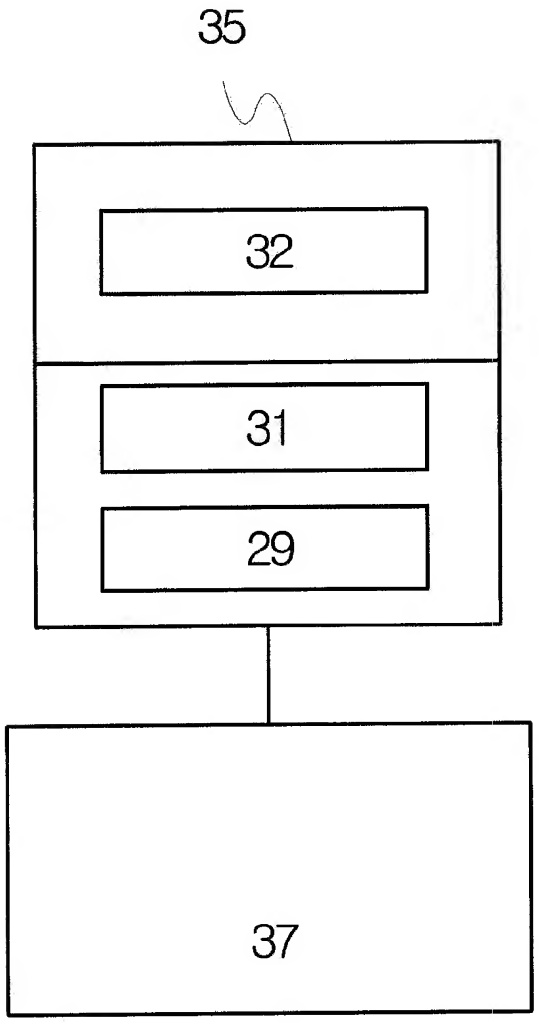
【도 1m】



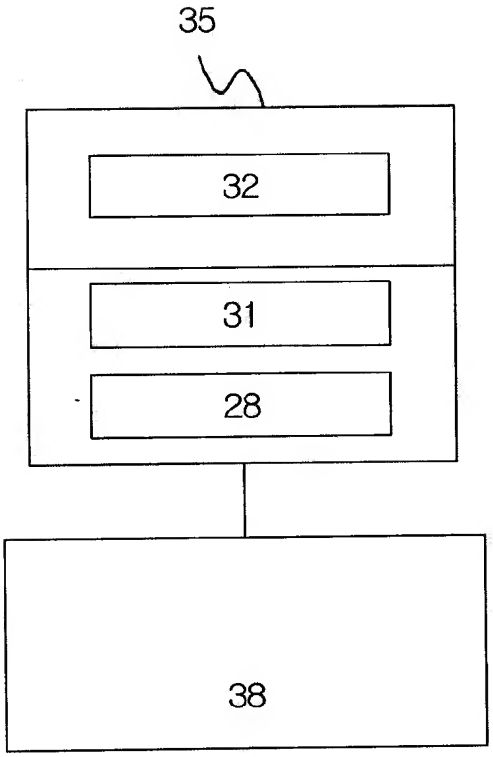
【도 1n】



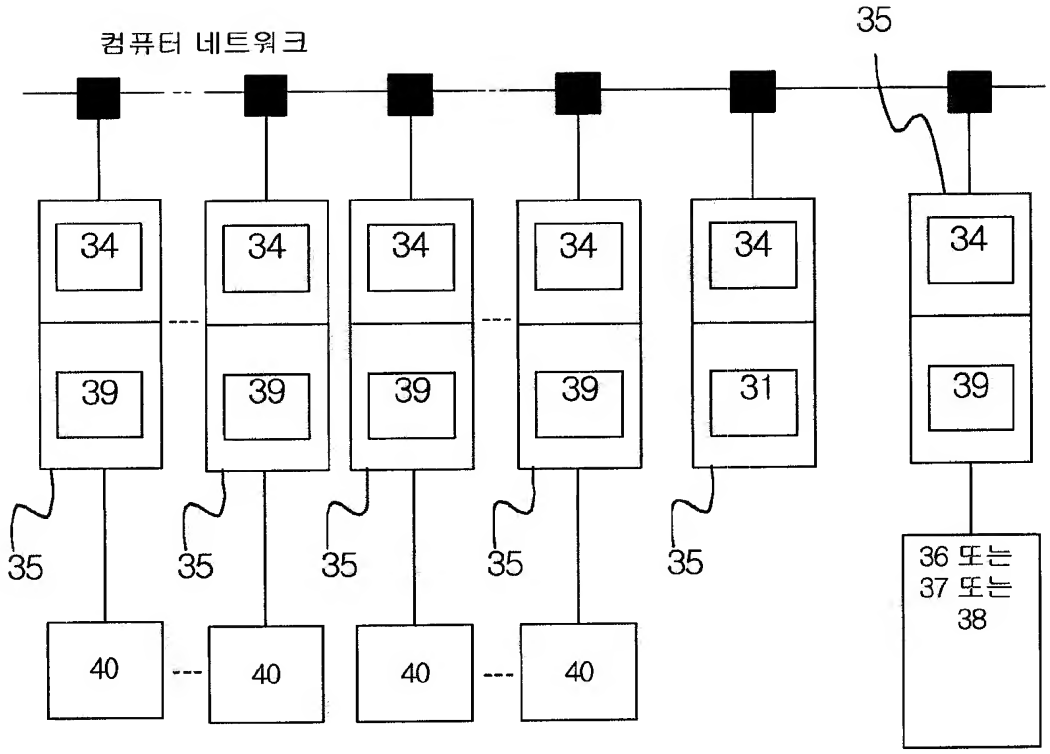
【도 1o】



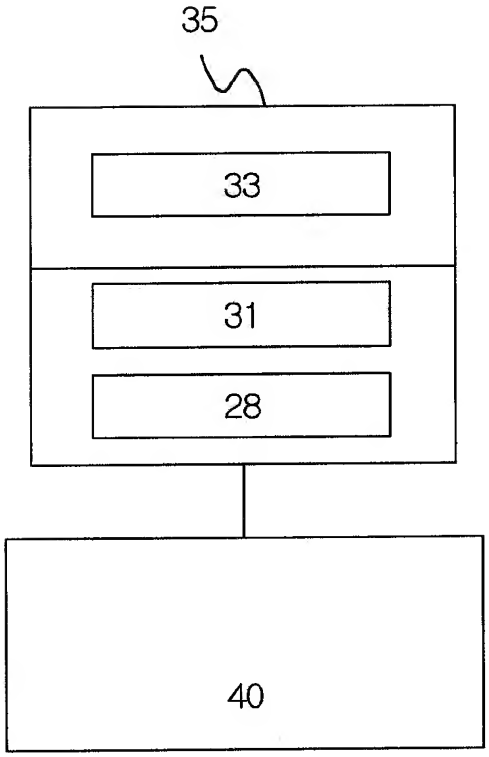
【도 1p】



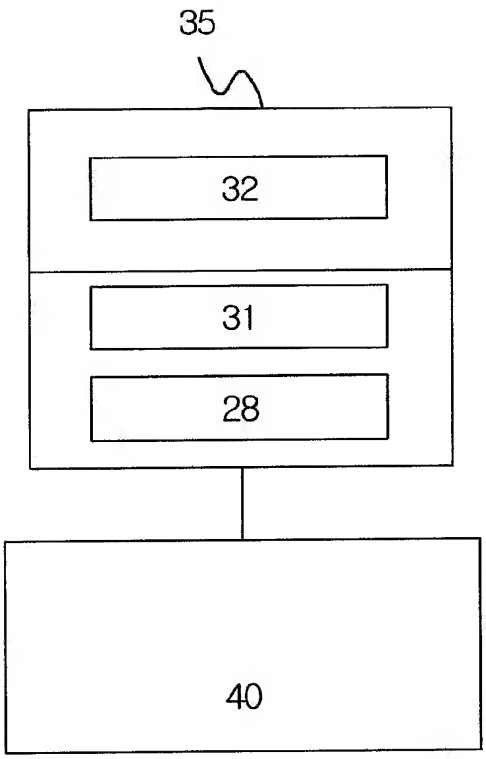
【도 1q】



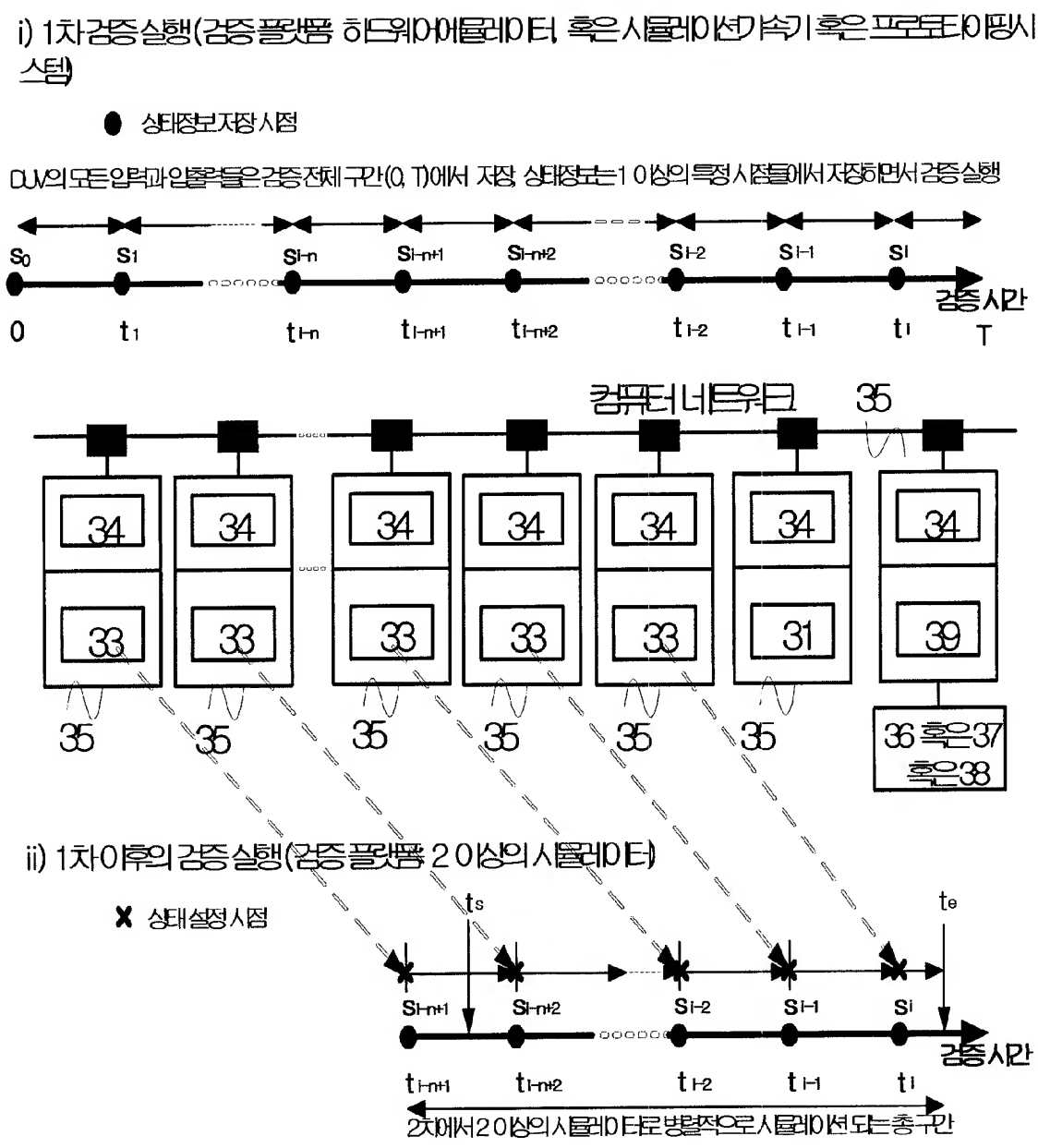
【도 1r】



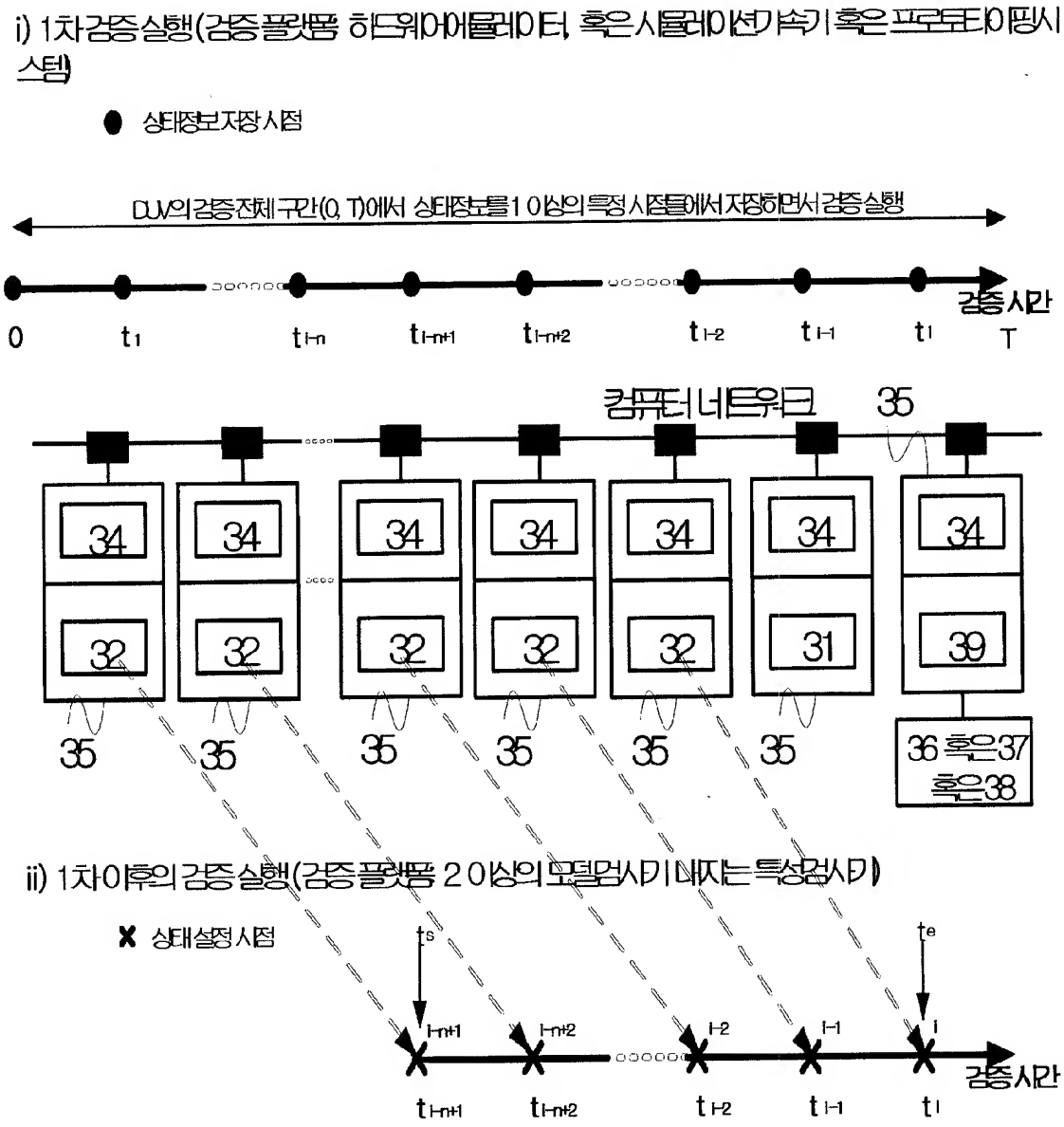
【도 1s】



【도 2】

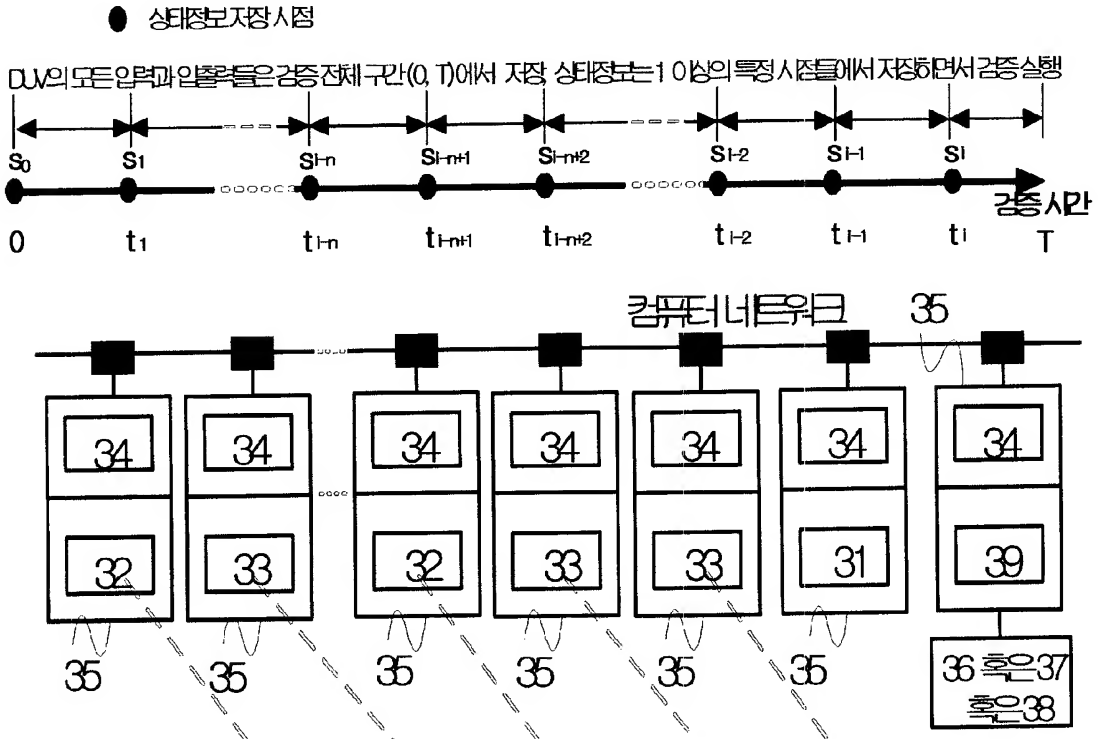

i) 1차검증실행(검증플랫폼 하드웨어에뮬레이터, 혹은시뮬레이션가속기혹은프로토타이핑시스템)
상태정보저장시점
DUV의모든입력과입출력들은검증전체구간(0, T)에서 저장, 상태정보는10상의특정시점들에서저장하면서검증실행
S0
S1
Si-n
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증시간
0
t1
ti-n
ti-n+1
ti-n+2
ti-2
ti-1
ti
T
컴퓨터네트워크 35
34
33
31
39
35
36 혹은37 혹은38
ii) 1차이후의검증실행(검증플랫폼 20상의시뮬레이터)
상태설정시점
ts
te
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증시간
ti-n+1
ti-n+2
ti-2
ti-1
ti
2차에서20상의시뮬레이터로병렬적으로시뮬레이션되는총구간

【도 3】

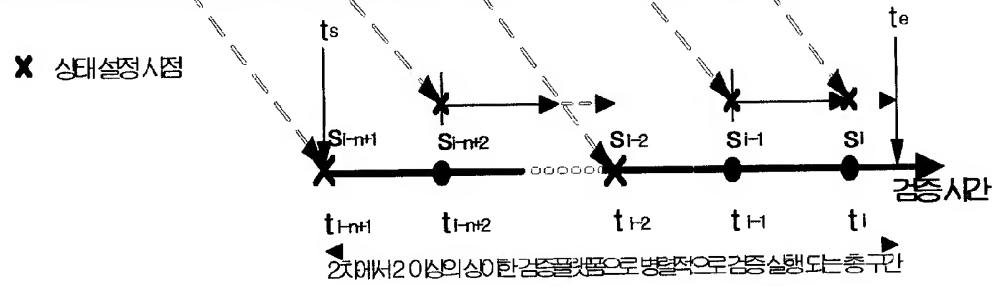

i) 1차검증실행(검증플랫폼 하드웨어에뮬레이터, 혹은시뮬레이션기록기혹은프로토타이핑시스템)
상태정보저장시점
DUV의검증전체구간(0, T)에서 상태정보를 1 0상의특정시점들에서저장하면서검증실행
검증시간
0
t1
tI-n
tI-n+1
tI-n+2
tI-2
tI-1
tI
T
컴퓨터네트워크
35
34
32
31
39
36 혹은37 혹은38
ii) 1차이후의검증실행(검증플랫폼 2 0상의모델엄사기 내지는특성엄사기)
상태설정시점
ts
te
I-n+1
I-n+2
I-2
I-1
I
검증시간

【도 4】

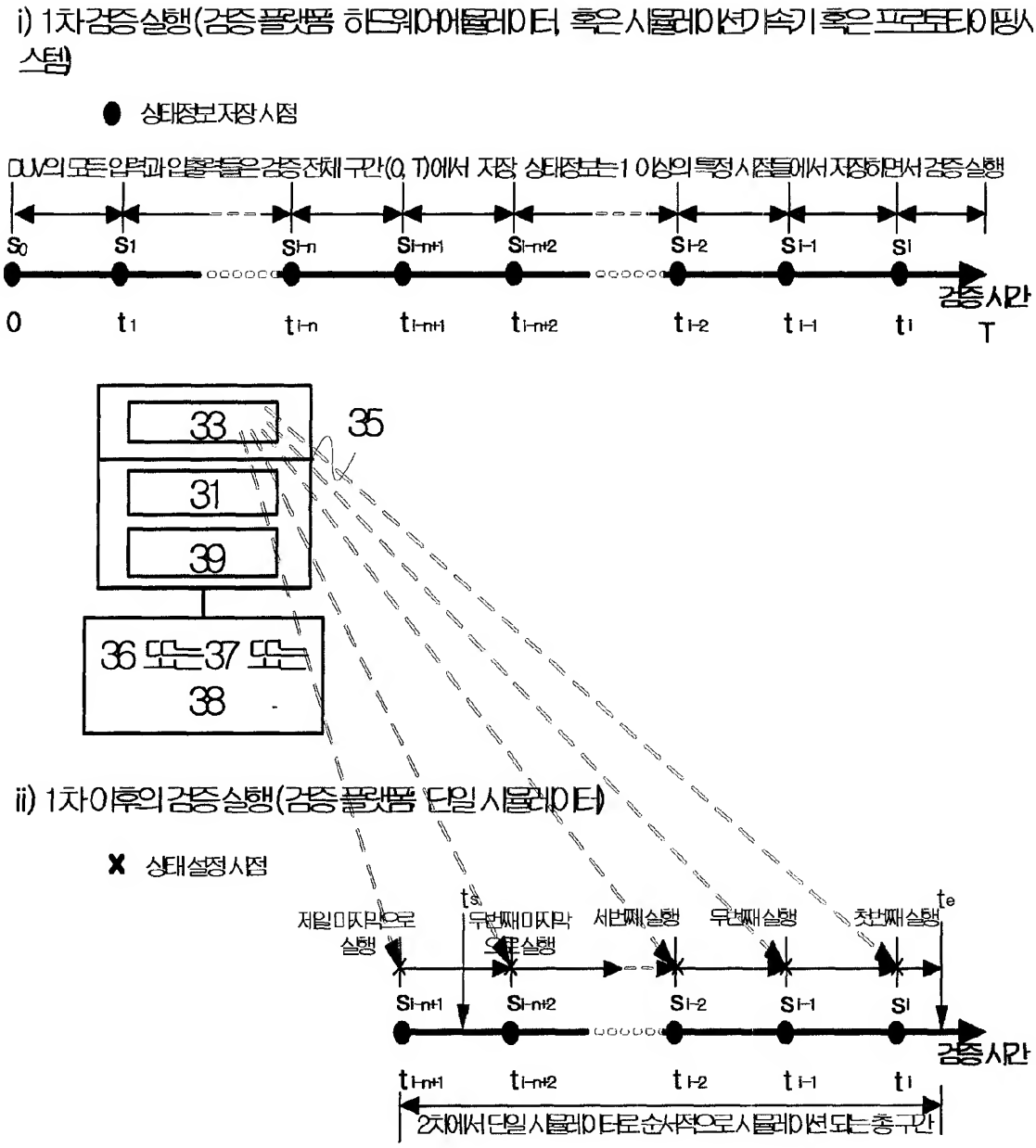
i) 1차검증실행(검증플랫폼 하드웨어에뮬레이터, 혹은시뮬레이션기속기혹은프로토타이핑시스템)



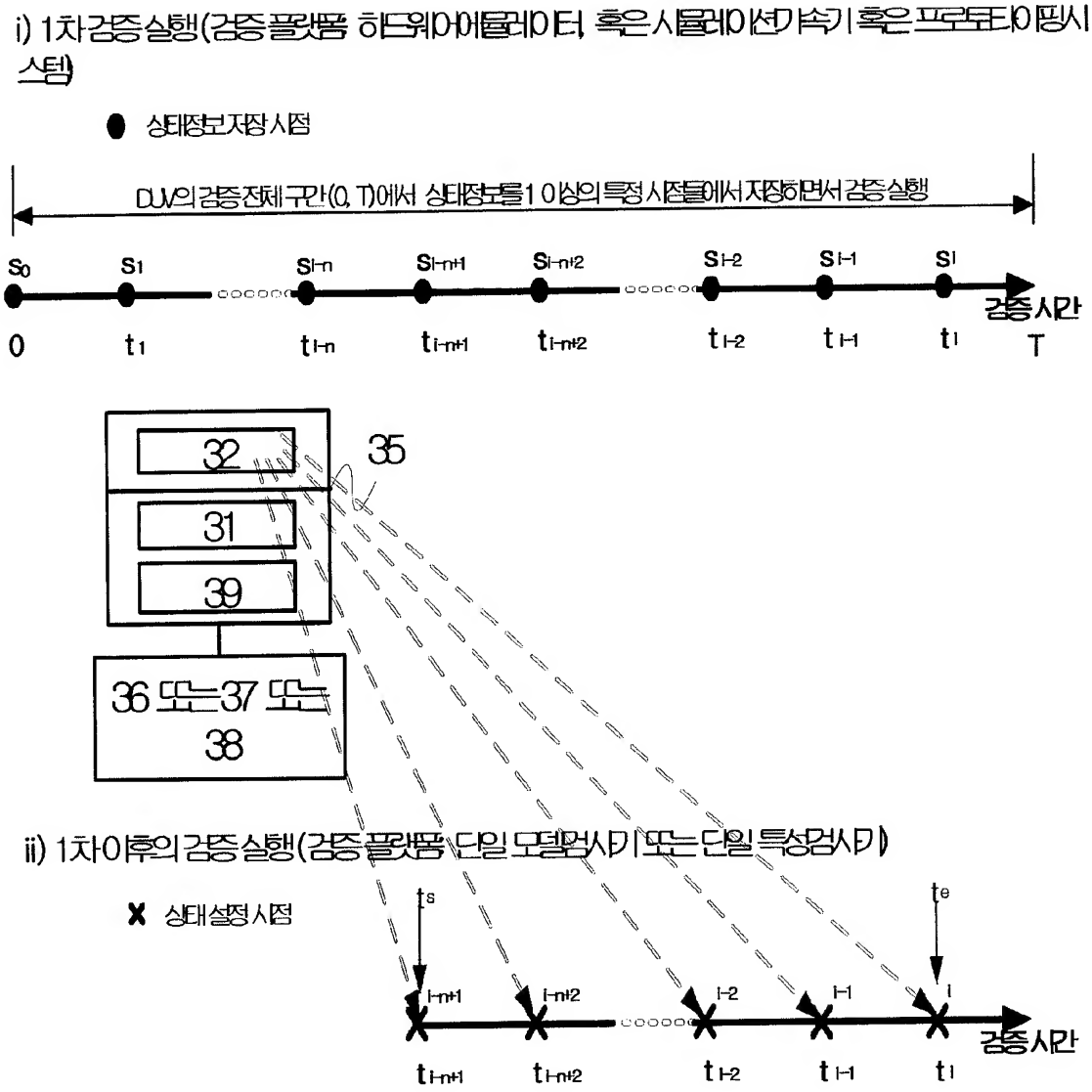
ii) 1차이후의검증실행(검증플랫폼 2 0이상의시뮬레이터와2 0이상의모델엄사기또는특성엄사기)



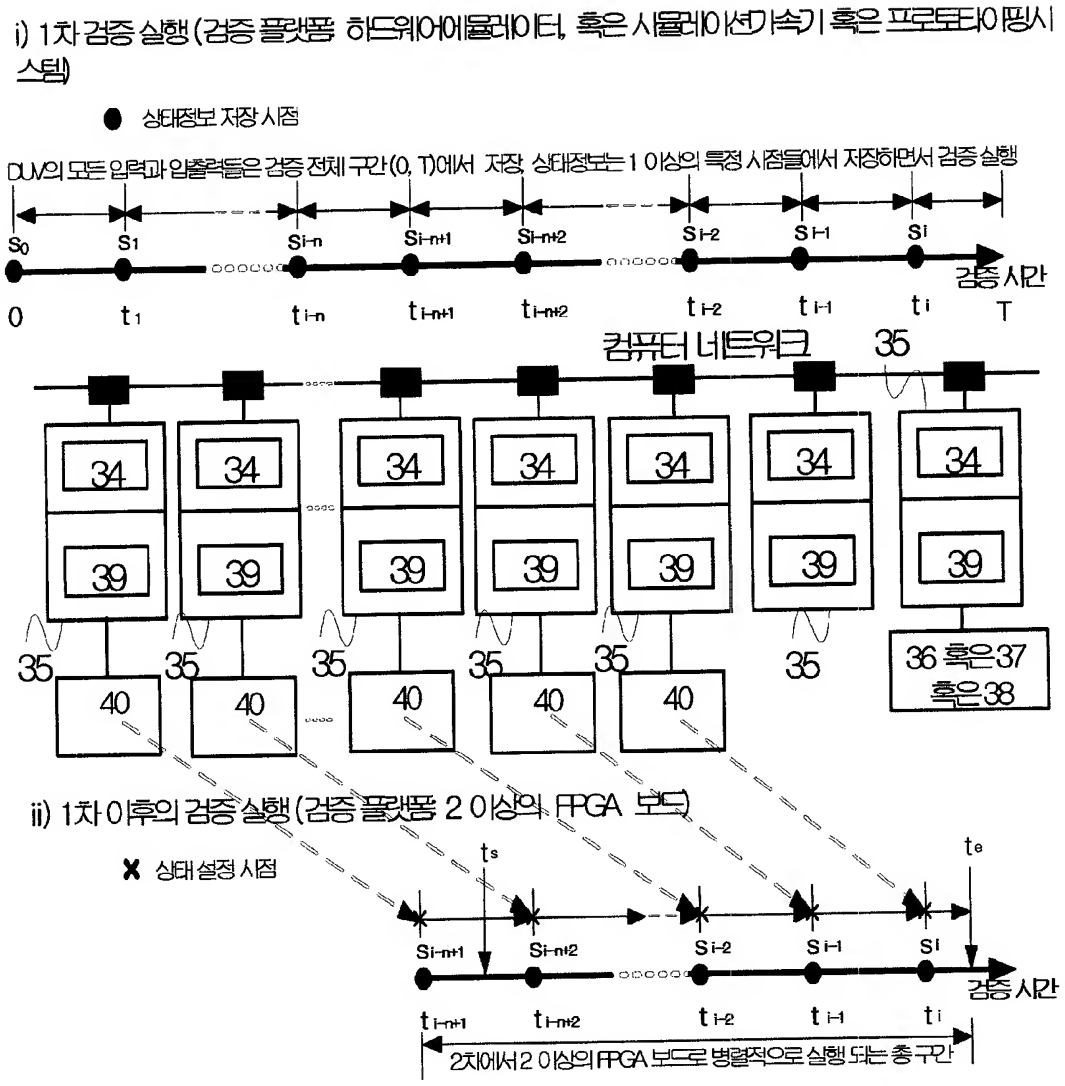
【도 5】


i) 1차검증실행(검증플랫폼 하드웨어에뮬레이터, 혹은시뮬레이션기속기혹은프로토타이핑시스템)
상태정보저장시점
DUV의모든입력과입출력들은검증전체구간(0, T)에서 저장, 상태정보는1 이상의특정시점들에서저장하면서검증실행
S0
S1
Si-n
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증시간
0
t1
ti-n
ti-n+1
ti-n+2
ti-2
ti-1
ti
T
33
31
39
35
36 또는37 또는
38
ii) 1차이후의검증실행(검증플랫폼 단일시뮬레이터)
상태설정시점
ts
te
처음이자막으로
실행
두번째이자막
으로실행
세번째실행
두번째실행
첫번째실행
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증시간
ti-n+1
ti-n+2
ti-2
ti-1
ti
2차에서단일시뮬레이터로순차적으로시뮬레이션또는총구간

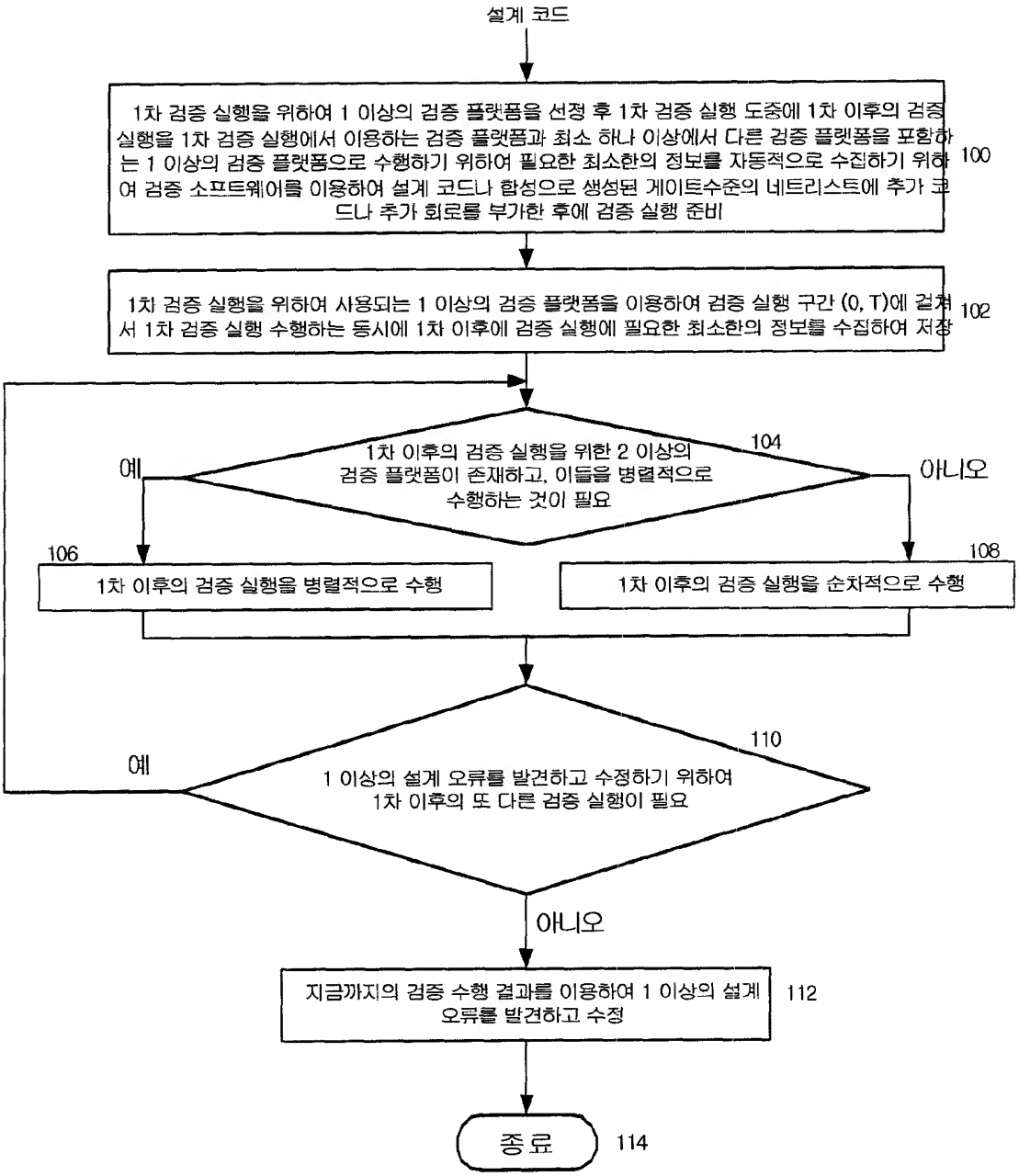
【도 6】


i) 1차검증실행(검증플랫폼 하드웨어에뮬레이터, 혹은시뮬레이션기속기 혹은프로토타이핑시스템)
상태정보저장시점
DUV의검증전체구간(0, T)에서 상태정보를1 0상의특정시점들에서저장하면서검증실행
S0
S1
Si-n
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증시간
0
t1
ti-n
ti-n+1
ti-n+2
ti-2
ti-1
ti
T
32
31
39
35
36 또는37 또는
38
ii) 1차이후의검증실행(검증플랫폼, 단일 모델멈사기 또는단일 특성멈사기)
상태설정시점
ts
te
i-n+1
i-n+2
i-2
i-1
i
검증시간
ti-n+1
ti-n+2
ti-2
ti-1
ti

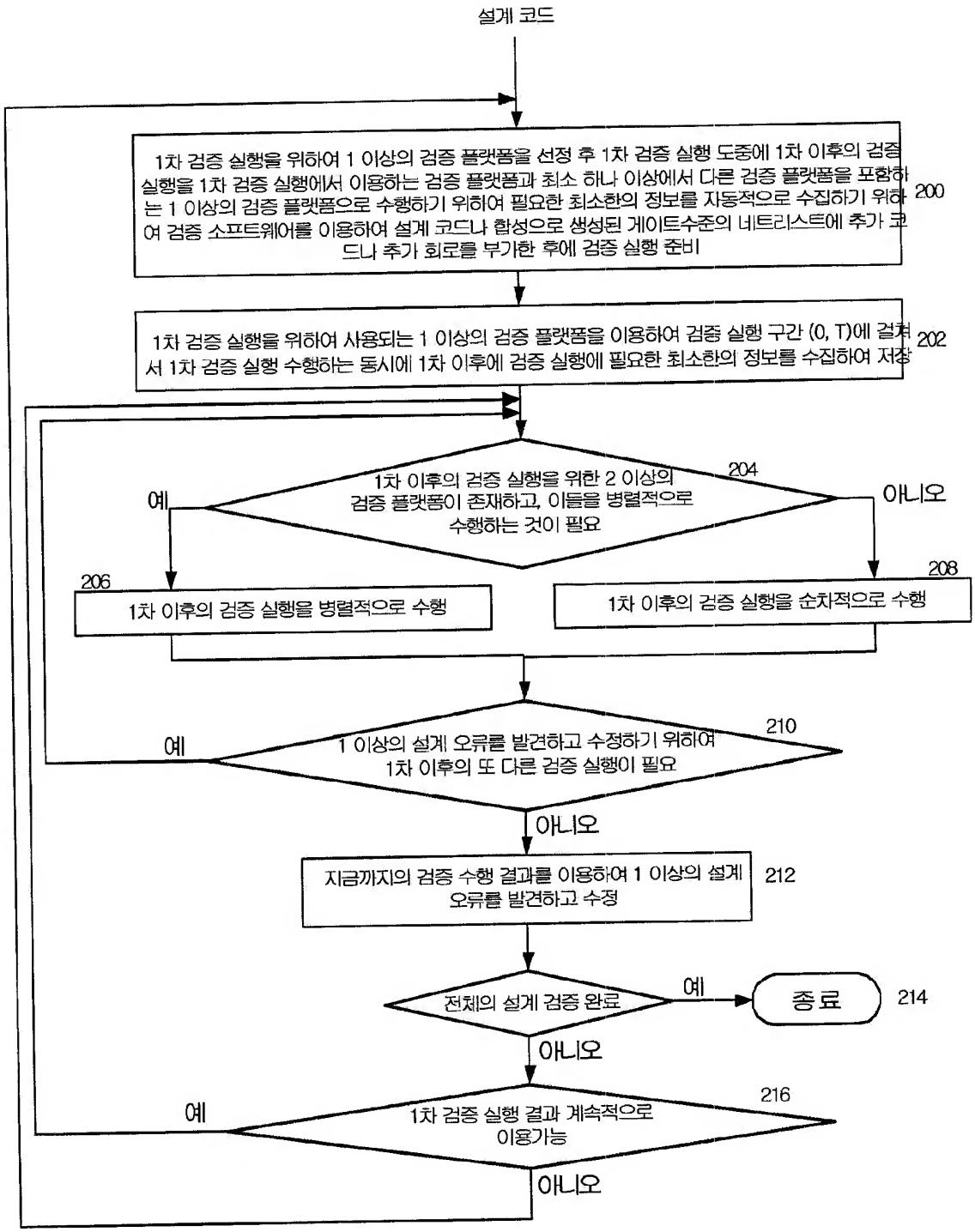
【도 7】


i) 1차 검증 실행 (검증 플랫폼: 하드웨어에뮬레이터, 혹은 시뮬레이션 가속기 혹은 프로토타이핑 시스템)
상태정보 저장 시점
DUV의 모든 입력과 입출력들은 검증 전체 구간 (0, T)에서 저장, 상태정보는 1 이상의 특정 시점들에서 저장하면서 검증 실행
S0
S1
Si-n
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증 시간
0
t1
ti-n
ti-n+1
ti-n+2
ti-2
ti-1
ti
T
컴퓨터 네트워크
35
34
39
40
36 혹은 37 혹은 38
ii) 1차 이후의 검증 실행 (검증 플랫폼: 2 이상의 FPGA 보드)
상태 설정 시점
ts
te
Si-n+1
Si-n+2
Si-2
Si-1
Si
검증 시간
ti-n+1
ti-n+2
ti-2
ti-1
ti
2차에서 2 이상의 FPGA 보드로 병렬적으로 실행 되는 총 구간

【도 8】


설계 코드
1차 검증 실행을 위하여 1 이상의 검증 플랫폼을 선정 후 1차 검증 실행 도중에 1차 이후의 검증 실행을 1차 검증 실행에서 이용하는 검증 플랫폼과 최소 하나 이상에서 다른 검증 플랫폼을 포함하는 1 이상의 검증 플랫폼으로 수행하기 위하여 필요한 최소한의 정보를 자동적으로 수집하기 위하여 검증 소프트웨어를 이용하여 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 추가 코드나 추가 회로를 부가한 후에 검증 실행 준비
100
1차 검증 실행을 위하여 사용되는 1 이상의 검증 플랫폼을 이용하여 검증 실행 구간 (0, T)에 걸쳐서 1차 검증 실행 수행하는 동시에 1차 이후에 검증 실행에 필요한 최소한의 정보를 수집하여 저장
102
1차 이후의 검증 실행을 위한 2 이상의 검증 플랫폼이 존재하고, 이들을 병렬적으로 수행하는 것이 필요
104
예
아니오
106
1차 이후의 검증 실행을 병렬적으로 수행
108
1차 이후의 검증 실행을 순차적으로 수행
110
예
1 이상의 설계 오류를 발견하고 수정하기 위하여 1차 이후의 또 다른 검증 실행이 필요
아니오
지금까지의 검증 수행 결과를 이용하여 1 이상의 설계 오류를 발견하고 수정
112
종료
114

【도 9】


설계 코드
1차 검증 실행을 위하여 1 이상의 검증 플랫폼을 선정 후 1차 검증 실행 도중에 1차 이후의 검증 실행을 1차 검증 실행에서 이용하는 검증 플랫폼과 최소 하나 이상에서 다른 검증 플랫폼을 포함하는 1 이상의 검증 플랫폼으로 수행하기 위하여 필요한 최소한의 정보를 자동적으로 수집하기 위하여 검증 소프트웨어를 이용하여 설계 코드나 합성으로 생성된 게이트수준의 네트리스트에 추가 코드나 추가 회로를 부가한 후에 검증 실행 준비
200
1차 검증 실행을 위하여 사용되는 1 이상의 검증 플랫폼을 이용하여 검증 실행 구간 (0, T)에 걸쳐서 1차 검증 실행 수행하는 동시에 1차 이후에 검증 실행에 필요한 최소한의 정보를 수집하여 저장
202
1차 이후의 검증 실행을 위한 2 이상의 검증 플랫폼이 존재하고, 이들을 병렬적으로 수행하는 것이 필요
204
예
아니오
206
1차 이후의 검증 실행을 병렬적으로 수행
208
1차 이후의 검증 실행을 순차적으로 수행
1 이상의 설계 오류를 발견하고 수정하기 위하여 1차 이후의 또 다른 검증 실행이 필요
210
예
아니오
지금까지의 검증 수행 결과를 이용하여 1 이상의 설계 오류를 발견하고 수정
212
전체의 설계 검증 완료
예
종료
214
아니오
1차 검증 실행 결과 계속적으로 이용가능
216
예
아니오